Radboud Repository

Radboud University Nijmegen

PDF hosted at the Radboud Repository of the Radboud University Nijmegen

The following full text is a publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/63057

Please be advised that this information was generated on 2017-12-06 and may be subject to change.

Bayesian methods for learning to learn

Een wetenschappelijke proeve op het gebied van de
Natuurwetenschappen, Wiskunde en Informatica

Proefschrift

ter verkrijging van de graad van doctor
aan de Katholieke Universiteit Nijmegen,
op gezag van de Rector Magnificus,
prof. dr. C.W.P.M. Blom,
volgens besluit van het College van Decanen
in het openbaar te verdedigen
op vrijdag 10 oktober 2003,
des namiddags om 1.30 uur precies,

door

Bart Jacob Bakker

geboren op 26 juli 1975 te Zevenaar

Promotor	:	prof. dr. C.C.A.M. Gielen
Co-promotor	:	dr. T. Heskes
Manuscript commissie	:	prof. dr. M.C.A. van Zuijlen prof. dr. R. Eisinga dr. N. Vlassis (UvA)

90-9017241-6

This research was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

Contents

Chapter 1

Introduction

Artificial intelligence (AI) covers a broad field of research, the role of which is growing in everyday life. The concept of AI (see e.g. [58, 78]) varies from topics like the emulation of human thinking and behavior, man-machine interaction, and data mining to reasoning aiming at correct performance in complex tasks. This thesis is concerned with the latter. Examples of this part of AI are chess computers, expert systems for medical diagnosis, and voice or finger print recognition in security systems.

In order to achieve the ambitious aims of AI, a large variety of techniques is used. One of these techniques is called *neural networks* (see e.g. [39]). Research in this field was originally inspired by the structure and function of the human brain for multiple reasons: the brain is flexible, it can 'learn' to solve complex problems from examples, it can adequately process incomplete or even conflicting information, and it is capable to give a good performance in a wide variety of complex tasks. Most neural network models that have been developed so far, reflect an abstract, simple version of the biological neural network. Abstract 'neurons' are modeled as mathematical units that receive numerical inputs (analog to the electrical inputs received by a biological neuron) and pass a numerical output (which depends on the received inputs) to another neuron. An example of such a model is the (multi-layered) perceptron, which will be described in more detail in Section 1.3.1. Rosenblatt [74] and Werbos [85] have proposed 'learning' algorithms that can update the parameters of such models so that after some time a desired task is performed correctly. The updates are driven by examples of 'correct behavior', generally in the form of a model input and a corresponding, correct model output. This type of 'learning from examples', which is one of the building blocks of this thesis, can be seen as a further specialization within the field of AI.

Over the past years, the field of neural networks has moved in two different directions. One part of the research in this field is concerned with modeling the functionality and biological complexity of the human brain. Here, the

complexity of the neuron models increases rather quickly, and researchers are concerned with incorporating the neurophysiological and anatomical observations in relation to the functionality of the neural network. Their aim is not primarily to achieve a particular functionality, but rather to understand human perception and behavior in terms of physiological and anatomical (sub)cellular properties. The other part of the neural network community is moving away from its original biological inspiration, and concentrates on finding learning methods that are able to deliver a desired functionality for applications. This branch of neural networks is often referred to as *machine learning*. At present, methods in machine learning become more and more dominated by the *probabilistic approach*. This approach aims to bridge the gap between human and artificial intelligence in its own way. Part of this gap is due to the fact that most computer software is written to 'think' in absolutes: the temperature is 15.7662 degrees Celsius, if the lawn is wet and the sprinklers have been off, it *must* have rained, etc. However, the human thinking process is very different: we say it is about 15 degrees Celsius, it probably rained last night, etc. Although this may seem not very precise, it seems to be a crucial and inevitable step to deal with incomplete, inaccurate information, as we will explain below.

Neural networks are able to learn from a set of examples, and have thus made an important step towards human-like reasoning. One of the disadvantages they still have in comparison with human learners can be summarized through the concept of 'overfitting'. When a neural network is presented with a (training) set of examples, the learning process will adapt its parameters to reproduce these specific examples as well as possible. However, this training set is usually a very limited selection of all possible examples, and each training example may be corrupted by noise (i.e., the examples are chosen to mimic the desired behavior as closely as possible, but may not be perfect). The result is often that the neural network converges (through learning) to a very precise, very specialized model that performs extremely well on the training set, but much worse under new, unknown circumstances. Human learning, however, results in a less precise 'model' which incorporates the 'obvious' aspects of the examples, but ignores the details. We see that in practice the 'human model' is better able to generalize, and therefore performs better.

Another disadvantage of neural network models is their weak compatibility with expert knowledge. Ideally, a model would be able both to exploit expert knowledge about the task it needs to perform, and to extend this knowledge through learning from new examples. Expert systems are, however, often unable to learn from new examples, whereas neural networks generally learn *only* from new examples. It may sometimes be possible to incorporate domain specific expertise in the architecture of a neural network, but this is seldom easy. Furthermore, it is still impossible to extend this knowledge through learning, since learning does not change the architecture of a network. Once more, all of this is no problem for a human expert. Take for example an art

student, learning to recognize paintings by van Gogh. On the one hand his teachers will provide him with expert knowledge about the use of colors and brush stroke technique, on the other hand he will learn his recognition task through experience. This student will most likely not remember his teachers original instructions literally, nor will he learn an exact technique to distinguish between a van Gogh and a Degas. However, in some 'fuzzy' way theoretical knowledge and experience combine in his head, and make him an adequate 'van Gogh detector'.

Such examples form an inspiration for the next step in artificial intelligence. We take the concept of learning from the field of neural networks, and combine it with the concept of probability theory. The latter can be seen as a bridge between the exact reasoning of computers and neural networks, and the more imprecise thought processes in the human brain. On the one hand, probability theory is exact: it defines probability distributions, which are well defined functions of a known set of parameters that can be stored with high precision in a computer. Computers experience no difficulty working with probability distributions: there exist well defined rules on how to calculate the probability of a model given a set of observations (that can be described by this model). There are also rules stating how to combine a distribution that defines the probability of an observation given a collection of expert knowledge, and a probability distribution based on a series of examples. Compared to reasoning without uncertainty, however, probabilistic reasoning comes one step closer to imprecise, human reasoning. The probabilistic approach still features exact values for data instances and model parameters, incorporated as the means of probability distributions, but now these values are paired with expressions of their (un)certainty, in the form of standard deviations and variances. This may be seen as an exact approximation of the fuzzy representation of models and knowledge inside the human brain. Our hope (and experience) is that this new way of expressing parameter values will eliminate some of the disadvantages of traditional AI compared to human thinking.

An example of the difference between reasoning without uncertainty and the probabilistic approach is the following. Consider the precise statement 'the train from Edinburgh to Inverness leaves at $t_E = 15.30$h'. In probabilistic terms this translates to

$$t_E \sim P(t_E|\hat{t}_E, \sigma_E^2)\,, \tag{1.1}$$

i.e., t_E is described by a probability distribution, with a mean value $\hat{t}_E (= 15.30\text{h})$ and a standard deviation σ_E (where for this example we will take $P(t_E)$ to be a Gaussian distribution). The latter indicates how much an actual measurement of t_E is expected to differ from $\hat{t}_E$. If we would take the train to Edinburgh each day, we expect it to leave at 15.30h on average, but we also expect that on average the difference between $t = 15.30$h and the actual departure time t_E is σ_E (minutes). One example of the added richness

of the probabilistic approach is the following: if a precise model of the British railway system would state that my train from London arrives in Edinburgh at $t_L = 15.28$h and that $t_E = 15.30$h, it also tells me that I will always catch my connection. The probabilistic approach however, will express both times in probability distributions and will tell me that I can expect to catch my train, but it will also tell me the probability that I may miss it. Based on this extra information I can either decide to take the risk, or to take an earlier train.

This thesis contributes to one of the current developments in AI: the combination of learning from examples, as has already been done for neural networks, and the use of probability theory. This combination opens up a whole new form of learning, and produces a wide array of robust, flexible models that outperform classical neural networks in many ways. This thesis concentrates on a part of probability theory that is well suited for learning: the Bayesian approach, which will be described in detail in the next section. The aim of this thesis is to extend existing (learning) methods through this approach. We will show that the overfitting problem can thus often be prevented, and that the Bayesian approach is an excellent way to incorporate expert knowledge in models. There is an enormous variety of models, in many different application areas, that can all benefit from an update through the Bayesian approach. This thesis implements such updates in four different area's, which will be described in Section 1.2.

1.1 Bayesian learning

Bayesian statistics is a part of probability theory that has become increasingly popular in the field of neural networks in the past decade. The theorem from which this part of statistics derives its name, is Bayes' theorem:

$$P(M|O) = \frac{P(O|M)}{P(O)} P(M) \,. \tag{1.2}$$

Let M represent some model of the world, and let $P(M)$ be the *prior belief* in this model. Bayes' formula now tells us what to do if our belief in the model M is put to the test, that is, when we observe some phenomenon O that can be described by our model. The fraction in Bayes' formula will be larger than one if the probability of the new observation under the current model ($P(O|M)$) is higher than its 'average' probability $P(O)$, and smaller than one otherwise. The average probability $P(O)$ can be expressed by $P(O) = \int dM\, P(O|M)P(M)$, an average over model predictions that is weighted with the belief in each model. So, if the model M is better able to describe the new observation O than the average model, our belief in M is strengthened, otherwise it is weakened.

A goal of Bayesian learning is to rate a model M through its ability to describe as many observations O as possible (within the boundaries of a specific learning problem). In practice we cannot consider all possible observations, so instead we use a finite set of data D, which is called a *training set*. The marginal probability of the training set, $P(D)$, is generally unknown and will be ignored throughout the learning process. The more interesting part of learning lies in finding suitable definitions for $P(D|M)$, also called the *data likelihood*, and $P(M)$, called the *prior distribution*. Note that the data likelihood is very different from the marginal data probability $P(D)$. The former defines the probability of an observation under a model M, and is therefore instrumental in the estimation of M's ability to describe observations. The latter is the probability of selecting the specific training set D from all 'available' observations, which does not play a role in the learning process.

In the full Bayesian approach it is enough to find expressions for the prior and the data likelihood. With these, we can construct the *posterior probability* $P(M|D)$ for any model M. The posterior can then be used to find the most probable model, or to generate predictions for new observations. Such observations often occur as a combination of a known input x and an output y that is to be predicted by the model. To predict y given x, we find an expression for

$$\begin{aligned} P(y|x,D) &= \int dM P(y|x,D,M)P(M|x,D) \\ &= \int dM P(y|x,M)P(M|D)\,, \end{aligned} \tag{1.3}$$

where in the second line we have left out D and x, since the probability of y no longer depends on the training data once the model M is given, and the probability of the model M is independent of any new input x. We can use the expectation value for y under this distribution,

$$\langle y \rangle = \int dy\; yP(y|x,D)\,, \tag{1.4}$$

as a prediction.

In the full Bayesian approach we set the prior $P(M)$ in advance and keep it fixed throughout the rest of the procedure. In the *empirical* Bayesian approach we may change this prior in the optimization process. Let us denote by Λ the

hyperparameters that describe the prior. We will find those values for Λ that are most likely, given the observations:

$$\begin{aligned} \Lambda_{\text{opt}} &= \underset{\Lambda}{\text{argmax}}\, P(\Lambda|D) \\ &= \underset{\Lambda}{\text{argmax}}\, P(D|\Lambda)\frac{P(\Lambda)}{P(D)} \\ &\propto \underset{\Lambda}{\text{argmax}} \int dM P(D|M)P(M|\Lambda)\,, \end{aligned} \tag{1.5}$$

where we expect each value for Λ to be equally likely *a priori* (although in principle it is possible to construct a so-called *hyper prior* distribution $P(\Lambda)$).

1.2 Concepts and application areas

A specific use of probability theory is the expression of the concept 'alike' in mathematical terms. Any human will use this concept intensively, throughout all aspects of everyday reasoning. Mostly, we call it experience: when we encounter a problem that is very much 'like' a problem we encountered and solved earlier, we just try the same strategy again. We may find that, since the problem is not exactly equal to the solved one, we have to change our strategy a bit, but the main idea will still be valid. It also works the other way around. When we solve similar problems quite often, we may start out by each time designing a strategy from scratch and reinventing the wheel a couple of times, but in time we will come to recognize a common denominator in all of our efforts, and use that 'average strategy' as a starting point for each new try.

Computers can be instructed to exploit similarities in learning tasks as well. This thesis will cover several situations where we want the computer to perform a collection of independent, yet similar tasks. The likeness of the tasks can be seen as a form of expert knowledge. This knowledge can be expressed in a Bayesian prior, which can subsequently be used to model collections of tasks better. In this thesis we will show both examples where the estimation of such priors leads to the detection of an average strategy, and examples where it uses the knowledge gained in learning one task to learn other tasks better.

1.2.1 Survival analysis

Survival analysis is a learning problem that can be described as a series of parallel tasks, where we want to know what chance a group of patients have to survive a particular length of time after the initial diagnosis of a serious, life-threatening disease. One may ask the question what the probability is to survive a period of five months, or two years, or three years. Existing methods

estimate these probabilities through observation of patients that survive more than five months, two years, or three years. Often, however, these questions were addressed by separate approaches since it was not acknowledged that the arguments and data which underlie the prediction tasks are 'alike', especially when predictions are related to periods close together in time. Subsequently, very erratic predictions could be produced. In Chapter 2 we introduce a prior distribution that implements the notion that predictions for periods of roughly similar duration should not differ strongly. A second prior distribution will be constructed to prevent that predictions for different patients with similar disease symptoms become *too* different. This prevents overfitting on the training data, which is a significant problem in the classical model.

1.2.2 Task clustering

Another example is the prediction of single copy newspaper sales at a number of different outlets. Although each outlet may have its own characteristics, the task of predicting newspaper sales is sure to be 'similar' for different outputs. We encode this belief in the form of a Bayesian prior distribution, as well as in the model structure for each prediction model. That is: one part in all of the models is the same for all models and captures the general common characteristics, whereas the other part is only 'similar' and provides room for differences between outlets because of outlet-specific differences. For the latter case, the similarity assumption is implemented by making (part of) each model subject to a common prior relative to an 'average model'. In this way, we only need to consider the similarity between a model for a particular outlet and the average model, and we do not have to make pair-wise comparisons between each possible combination of two models.

The created 'link' between similar tasks leads to a stronger generalization ability for each model. Although we allow different models for different tasks, each model can make use of the entire set of training data, over all tasks. Since each model is thus optimized on the full database instead of on the (much smaller) task specific data set, the risk of overfitting becomes much smaller. We could also say that the models 'learn from each other'. From this viewpoint, the link between models serves as a conduit to pass knowledge that is learned for one model on to another model.

Sometimes, there is information that some outlets are bound to be 'more similar' to each other than to other outlets: newspaper sales at the beach and in recreational areas may well conform to other rules than sales in big cities. This can be expressed in Bayesian priors. Instead of one average model, we allow multiple 'local average' models. Each outlet's model will then be 'assigned' to the average model that it resembles most, along with its fellow models that it is 'most similar' to. Not only are models better able to perform

their prediction tasks in this way, we also learn more about the structure of the tasks by looking at the emerging 'task clusters'.

1.2.3 The hierarchical dynamical model

Survival analysis features a relation between predictions for neighboring points in time. The same can be done for newspaper sales, because this week's sales figures may well tell us something about next week's sales. In this case we apply a concept from the world of time series analysis, called a dynamic linear model. We presume that sales for any particular outlet is a function of some 'hidden state' which evolves through time. In this model each state is expected to be similar, but presumably not identical, to the previous state (the state corresponding to the same outlet, but to an earlier time) or a known transformation thereof. Apart from that, we hold on to the notion that all outlets behave more or less similar to each other. Hidden states for different outlets, yet for the same point in time, are linked to one 'average state'. Each state is assumed to be drawn from a distribution that depends both on the previous state, which is different for each outlet, and on the common factor that is implemented through the average state.

1.2.4 Ensemble clustering

When tasks do not naturally occur as an ensemble of similar tasks, we can still make them so. In the field of neural networks there are many methods that produce an ensemble of similar models. One such method is called *bootstrapping* (see e.g. [30]): instead of training one model on one complete database, an ensemble of models is trained, each on a slightly different subset of the complete database. In this way, we represent one task (learning on the complete data set) as a collection of similar tasks (learning on many subsets of the complete data set). Another example lies in the Bayesian approach to learning itself. Here, instead of learning one model, we set up a probability distribution over infinitely many possible models. Sampling from this distribution (see Section 1.3.3) yields a large (but finite) ensemble of similar models, even when an ensemble of similar tasks cannot be specified explicitly.

Ensembles of models may present an accurate, balanced approach to the learning task at hand, but it is often hard to keep a good overview of the characteristics of the models. It would be easier if some sort of compact summary was available to capture all of the model characteristics in a manageable format. In Chapter 5 we will provide such a summary, in the form of *cluster centers*. The ensembles of models are divided into clusters, where models inside one cluster are more 'like' each other than models in two different clusters. Each cluster is represented by a 'representative model', or cluster center. We will show that a representation of the full ensemble of models by a few cluster

centers makes the ensemble more manageable. Yet, this reduced ensemble of cluster centers is still able to describe the data as well as the full ensemble.

1.3 Models, methods and approximations

Each chapter in this thesis describes a model, chosen to perform a specific task, and a posterior distribution. The latter assigns a probability to the model, based on the data in a training set. The model, and therefore the posterior distribution, have a different form in each chapter. The posterior often has such a complex structure that it cannot be easily expressed as a function of the hyperparameters. It generally involves a high-dimensional integration that cannot be done exactly, i.e. there exists no general analytical expression for its result. In this case we have to resort to approximations to the exact distribution. The design and implementation of such approximations cover in fact the larger part of the work described in this thesis. The remainder of this section gives a description of a variety of models and approximation methods, and describes how they are implemented in this thesis.

1.3.1 The multi-layered perceptron

Throughout this thesis a model structure called a 'multi-layered perceptron' (MLP) will often be used to give predictions for observable outputs given certain inputs. The architecture of such a model is depicted in Figure 1.1. The circles at the bottom of the picture represent the inputs to the model. The inputs are connected to the middle layer of circles which we call 'hidden units', since they correspond to values, used within the model, that do not represent observable inputs or outputs. Each hidden unit is connected to all of the inputs, indicating that its value depends on a weighted sum of the inputs:

$$h_i = g\left(\sum_j W_{ij} x_j + b_1\right) , \tag{1.6}$$

where the 'weights' W_{ij} are represented by the connecting lines, and are called the 'input-to-hidden weights'. The parameters h_i represent the hidden unit values, b_1 represents the bias (an added constant) and the parameters x_j represent the inputs to the model.

The top layer of circles represents the outputs of the model, which depend on a weighted sum of the values of the hidden units:

$$y_k = f\left(\sum_i V_{ki} h_i + b_2\right) , \tag{1.7}$$

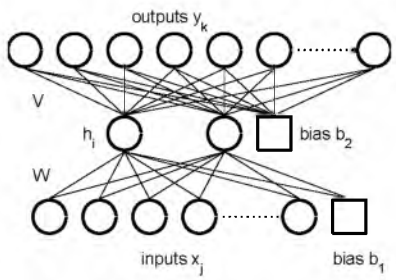


Figure 1.1: The multi-layered perceptron.

where the parameters y_k represent the model outputs and the parameters V_{ki} represent the hidden-to-output weights.

In a linear MLP the transformation functions $g(.)$ and $f(.)$ 'do nothing': $g(z) = f(z) = z$. In many cases, however, we will use a nonlinear MLP with nonlinear transformation functions, such as the hyperbolic tangent. Nonlinear MLPs feature a more complex input-output relationship than linear MLPs, and are therefore able to model more complex functions.

The number of hidden units will be lower than the number of inputs for MLPs in this thesis. In this way a lower-dimensional representation of the inputs is realized. This leads to the automatic detection of (important) features in the data, and to better generalization [7, 23].

1.3.2 The proportional hazards model

The most frequently used model in survival analysis is the proportional hazards model. This model expresses the probability for a patient to survive up to a certain time as a simple product of two functions. The first function, called the proportional hazard, depends on the inputs describing the patient (age, weight, etc.) but not on time. The second function, called the baseline hazard, depends only on time and is therefore the same for all patients.

The baseline hazard is classically estimated as follows. Time is divided into a number of discrete intervals. The estimated baseline hazard for each interval is calculated as the number of patients (followed in the underlying medical studies) who pass away in this interval, divided by the number of patients who are alive at the start of the interval. Random fluctuations may lead to a very 'wild' variation of the baseline hazard through time, especially when the number of patients included in the studies is rather limited.

In Chapter 2 we implement a variation of the proportional hazards method in the form of a nonlinear MLP. We impose two priors on the model parameters. One prior serves to prevent the baseline hazard from becoming too wild. The

other prior acts on the proportional hazard, and (again) serves to prevent overfitting on the training data.

1.3.3 Sampling the posterior

A probability distribution can be represented by a collection of samples. Take for example the distribution $P(M|\Lambda)$ over possible models M. We can select a number of different models M_i. If the probability that we choose a specific M_i is given by $P(M_i|\Lambda)$, we are said to *draw samples* from $P(M|\Lambda)$. Sampling can e.g. be used to approximate an integration: when we have drawn a (large) number of candidate models M_i (defined through their model parameters) from $P(M|\Lambda)$, we can approximate $P(\Lambda|O) \propto \int dM P(O|M)P(M|\Lambda)$ by computing $P(O|M_i)$ for each of our samples, and taking the average. This approach is in fact a popular form of *numerical integration.*

Some probability distributions (e.g. Gaussian or uniform) are very easy to sample from. Those encountered in Bayesian learning, however, are often more complex, and more involved procedures are required. One such procedure is *importance sampling.* Here we draw samples from a standard (e.g. Gaussian) distribution that may resemble the more complex original. Each draw is accepted if its probability under the complex distribution ($P(s|C)$) exceeds its probability under the alternative ($P(s|G)$). If not, it is accepted with probability $P(s|C)/P(s|G)$ and otherwise discarded. In practice it is often very hard to find a standard function $P(s|G)$ that closely resembles the distribution we wish to sample from. The result is that many samples will have a very low ratio $P(s|C)/P(s|G)$. Such samples have a high probability to be discarded, and are likely drawn for nothing. Markov chain Monte Carlo (MCMC) sampling, which does not have this disadvantage, will be discussed in Chapter 2 .

1.3.4 Approximate distributions

Sampling from a posterior distribution is always possible and, in the limit of infinitely many samples, yields exact results for all expectation values. It does however take a lot of time to get a large enough number of samples, and subsequent calculations can be performed more elegantly when the actual posterior distribution function is available. When this distribution is too complex to be used directly, we often use an approximate distribution. This distribution will generally have a simpler form (e.g. Gaussian), and can easily be used for all further calculations. The approximation must of course resemble the exact distribution as closely as possible. A common measure of how 'close' two distributions are, is the Kullback-Leibler (KL)-divergence:

$$\mathrm{KL}(P(.), Q(.)) = \int dx P(x) \frac{\log P(x)}{\log Q(x)}\,, \tag{1.8}$$

where $P(.)$ and $Q(.)$ are the two distributions and x is the parameter the probability of which is evaluated. The KL-divergence is zero for $P = Q$ and positive otherwise. The optimal approximate distribution is obtained through minimization of its (KL)-divergence with the exact posterior distribution.

A much simpler way to estimate an approximate distribution is the Laplace approximation. The approximating distribution $P(x)$ in this case is assumed to be Gaussian with a mean $\bar{x}$, which is the choice for the parameter x that maximizes the exact distribution. Its covariance is related to the second derivative (with respect to x) of the log of the exact distribution, taken at this maximum:

$$\sigma^{-2} = \left[-\frac{\partial^2}{\partial x^2} \log P(x) \right]_{x=\bar{x}} . \tag{1.9}$$

This method is often not very accurate, yet it may require much less computation time.

The posterior distribution $P(x)$ in Chapter 2 is too complex to be used directly. We, therefore, propose a variational method where we approximate the posterior with a simple distribution $Q(x)$, and we minimize the KL-divergence $\mathrm{KL}(P(x), Q(x))$. We compare this approximation to MCMC sampling and to the Laplace approximation. We calculate expectation values for the model outputs under each approximation, and we compare the three methods in terms of prediction accuracy.

1.3.5 Input analysis and the Bayes factor

One of the tasks of each model in this thesis is to predict a new (i.e. not used for training), unobserved output based on a set of new, observed inputs. Some of these inputs may play a crucial role in this prediction, others may be totally unrelated to the output. Inputs that do not contribute to the correct prediction of a new output, may well hinder the learning process: they 'confuse' the model, and make it harder to find the 'true' dependence on the other inputs. It may therefore be important, especially when many inputs are present, to distinguish useful inputs from 'junk inputs'. An elegant method to purge the model from irrelevant inputs is related to the *Bayes factor*. This factor is defined as

$$BF = \frac{P(D|H_0)}{P(D|H_1)} , \tag{1.10}$$

the ratio of the probability of the (training) data given a certain null-hypothesis and the probability given another hypothesis. For the determination of the relevance of a certain input, we define the null-hypothesis as: the input is irrelevant, and can be removed from the model. The alternative hypothesis is the opposite: the input should stay. A high Bayes factor thus gives a strong indication that the considered input is in fact irrelevant.

In Chapter 2 we use the Bayes factor to get rid of all 'junk inputs', and show that this has a positive effect on the model's performance.

1.3.6 Multitask learning

Multitask learning is a concept that exploits the fact that models trained on similar tasks must be similar to each other. This can be implemented both through 'hard sharing' and 'soft sharing' of parameters. Hard shared parameters are incorporated in the model structure itself. The multi-layered perceptron has a suitable form for such an incorporation. Let each of the MLP's outputs represent the output for one task, and let the number of hidden units be much smaller than the number of outputs (see also Figure 1.1). It can easily be seen that, although each output (task) has its own set of hidden-to-output weights, all tasks share the same input-to-hidden weights. In fact, all tasks use the same lower-dimensional representation of the inputs, but they have their own final transformation of this representation.

Soft sharing can be implemented through a common prior distribution on (part of) the parameters of each model. This prior distribution can be a single Gaussian, implying that all soft shared parameters are scattered around the same mean set of parameters, or it can be more complex. A sum of Gaussians, for example, can be used to allow for multiple clusters of tasks. Each cluster will be characterized by the mean and variance of one of the Gaussians in the sum. Each parallel task will be modeled to have a particular probability to belong to any one of the clusters, determined in part by the observations corresponding to that task. This clustering of tasks leads to more accurate predictions of future observations, and offers a new insight in the characteristics of each task.

In Chapter 3 we implement both hard sharing and soft sharing of parameters, and study the effects of both the single Gaussian prior and clustering.

1.3.7 Graphical models

Graphical models are often used to present complex systems of interacting variables. Consider the model depicted in Figure 1.2. The open circles (nodes) in this picture represent hidden states of the model. Hidden states are vectors that describe local parts of a modeled system. Some of the hidden states are connected to each other, indicating conditional independencies. Take for example the three states $\boldsymbol{\theta}_2$, $\boldsymbol{\theta}_3$ and $\boldsymbol{\theta}_4$ in Figure 1.2. The states $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$ are connected in the graph, as are the states $\boldsymbol{\theta}_3$ and $\boldsymbol{\theta}_4$, but not $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_4$. This means that if $\boldsymbol{\theta}_3$ is known, the probability of $\boldsymbol{\theta}_4$ is independent of the state $\boldsymbol{\theta}_2$, or $P(\boldsymbol{\theta}_4|\boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = P(\boldsymbol{\theta}_4|\boldsymbol{\theta}_3)$. Such independencies make it possible to write down relatively simple expressions for $P(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N)$, the joint probability of all of the states in the network.

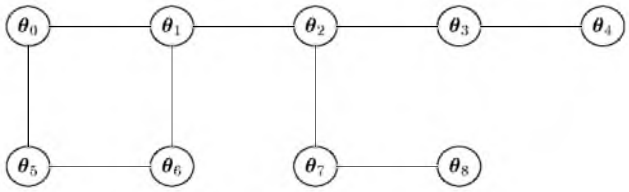


Figure 1.2: Example of a graphical model. The open circles represent hidden states, the connections indicate conditional independencies.

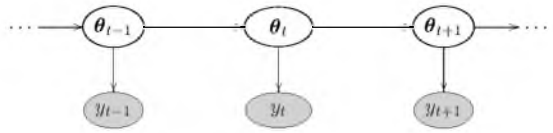


Figure 1.3: Example of a directed graphical model. The shaded areas represent observations, the open ellipses represent hidden states.

Time series structures are often represented as directed graphical models. In a *directed* graph (Figure 1.3) the nodes are connected by arrows. When for example there is an arrow from state $\boldsymbol{\theta}_t$ to $\boldsymbol{\theta}_{t+1}$, it means that we have an expression for $P(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$, the probability of the future state given the current state (but not necessarily for the joint probability $P(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1})$). A graphical model that features such independencies is said to have *Markovian* dynamics. That is, the state of the system at time t ($\boldsymbol{\theta}_t$) depends only on the previous state, $\boldsymbol{\theta}_{t-1}$. The hidden states can be connected to observations, represented by shaded ellipses. The connection signifies, as before, that the probabilities of these observations (at the same times t) depend on the hidden states that they are connected to. This type of graphical model is often used to describe time series.

1.3.8 The Kalman filter

When the states of a graphical model have continuous variables, their dependence on the previous state is often modeled through a *Kalman filter*. That is, the prediction for the current state reads

$$\boldsymbol{\theta}_t = A\boldsymbol{\theta}_{t-1} + \boldsymbol{\nu}_t \,, \tag{1.11}$$

a linear function of the previous state (through the *evolution matrix* A) with added Gaussian noise $\boldsymbol{\nu}_t$. A time series model that is determined by this type of evolution equations is called a dynamic linear model (DLM). Efficient methods exist to calculate the probability distribution $P(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T|y_1, \ldots, y_T)$ over the hidden states $\boldsymbol{\theta}_t$ under the above dynamics and given the observations connected to these states (as in the previous subsection).

In Chapter 4 we consider a collection of parallel time series. Each time series is modeled through hidden states $\boldsymbol{\theta}_{it}$, the dynamics of which are described by the Kalman filter equation. We also introduce an 'average' DLM which is modeled through hidden states $\mathbf{M}_t$. Each state $\mathbf{M}_t$ is connected to all of the hidden states $\boldsymbol{\theta}_{it}$ at the same time t. This model, therefore, features two types of dependencies: those through time, between subsequent states within one DLM, and those between an average state $\mathbf{M}_t$ and a parallel state $\boldsymbol{\theta}_{it}$. This type of model is an extension of the 'standard' dynamic hierarchical model (DHM). The standard DHM features connections between subsequent average states $\mathbf{M}_t$, and between states $\mathbf{M}_t$ and states $\boldsymbol{\theta}_{it}$. It does not, however, include dependencies through time between the states $\boldsymbol{\theta}_{it}$ of the parallel DLMs. The combination of dependencies through time *and* between the states $\mathbf{M}_t$ and $\boldsymbol{\theta}_{it}$ make calculation of the posterior very complicated. In fact, it can no longer be done within reasonable time. Therefore, in Chapter 4 we describe two approximations, which are compared to the exact model and to each other.

1.3.9 Soft clustering

Clustering methods have been around for a long time, and in many varieties. Most methods take a collection of objects (pieces of fruit, complex numbers, people, models) and sort them neatly in a fixed number of groups. Each object then belongs to exactly one cluster. In this thesis we will consider a 'softer' alternative where, instead of 'hard' assignments of one object to one cluster, objects will have a probability distribution over assignments, indicating 'how strongly' they belong to any cluster.

In Chapter 5 this method is applied to models. We define a distance measure between models that is based on model outputs. Each cluster is defined by its cluster center and the assignment probabilities for each model depend on its distance to the corresponding centers. The most important role of these centers, however, is to give a compact representation of the full ensemble of models. The results are used to gain insight in the workings of bootstrapping methods, and to find meaningful clusters in a real-world multitask learning problem.

1.4 Conclusions and discussion

In Chapter 2 we show that survival analysis can benefit from the Bayesian approach. The overfitting problems involved in the original proportional hazards model are prevented in this approach, and our predictions are better than those of the classical model. That is, both models estimate a probability for a new patient (who was not involved in training the model), to pass away after a certain time t. When we compare these estimates for time $t_\dagger$,

the time that the patient is observed to pass away, the estimated probability of this observation is (on average) up to 80% higher under our model than under the classical model. Comparison of the three approaches to the corresponding Bayesian posterior, i.e. MCMC sampling, a variational approach and the Laplace approximation, reveals that whereas MCMC sampling yields the best predictions (up to 30% more accurate than the other two approaches), the variational approach is better suited to calculate the Bayes factor. This factor can be calculated analytically when the posterior is expressed as an analytical function, but requires a very large number of samples to be estimated numerically. Application of the Bayes factor enabled us to eliminate over 85% of the model inputs, without any loss of functionality. In fact, the thus reduced model, which is no longer hampered by 'irrelevant' inputs, performs significantly better (up to 5%) than the model with all inputs present.

In Chapter 3 we present a Bayesian version of multitask learning. Test results are obtained (among others) on a database describing single copy newspaper sales in the Netherlands. We show that multitask learning with one cluster (i.e. with a single Gaussian prior) offers better predictions than single task learning, where a model is optimized for each task separately. If we define the prediction error as the difference between the predicted value and the actual number of newspapers sold, the error through multitask learning was on average 10% smaller than the error through single task learning. Bayesian multitask learning also beats (by 1%) the non-Bayesian variant, where only hard sharing is implemented: all tasks are predicted through one MLP with a separate output for each task. Task clustering (where we implement a prior distribution that is a sum of Gaussians) does not improve performance on the newspaper data. However, it does yield a meaningful clustering of tasks: sales at outlets in touristic areas and small villages are assigned to one cluster, outlets in relatively large cities to another. Chapter 3 demonstrates the two uses of task clustering: first, it provides better predictions (sometimes already for one cluster), second, it provides new insight in a task's characteristics through its cluster assignment.

Chapter 4 introduces the combination of multitask learning and time series analysis in the form of a dynamic hierarchical model. The resulting model is shown to outperform the standard DHM on which it is based. A possible reason for this improvement can be seen from a plot of the latent state means through time: where the means for the parallel time series behave erratically for the old model, the dynamics for the means inferred through the model described in this thesis are much more smooth. Two approximations (variational and factorial) to the full graphical model are studied in this chapter. In terms of prediction quality, both perform equally well on the considered data. The computation time that is required to calculate the posterior distribution through either approximation grows linearly with the number of parallel tasks, whereas the growth for exact inference is much stronger. The factorial

approach is slightly slower than the variational approach, but also more accurate: in terms of the KL-divergence it approximates the exact posterior more closely. This does however not translate to more accurate predictions since both approximations already predict nearly as well as the exact model. Both approximations are shown to feature exact means.

In Chapter 5 we present a model clustering method. We use this method to represent ensembles of models by a small number of cluster centers. The first set of models is obtained through bootstrapping. We show that predictions based on just a few cluster centers can be just as good as predictions based on the full ensemble. Model clustering is further used to demonstrate the effects of both bias and variance reduction in bootstrapping methods. The second ensemble is formed through independent learning of a collection of parallel tasks (one or more independent models are learned for each task). Here, model clustering is presented as an alternative form of multitask learning and task clustering. The clusters of tasks are shown to have similar characteristics to those found in Chapter 4. Both experiments show that model clustering can be used as an effective analytical tool for data mining.

Science may proceed, yet it is never finished. Although the work done for this thesis provides solutions for some problems, and improvements in some fields, it should also lead to new ideas for competing, alternative approaches. For example, the survival analysis method that is presented in Chapter 2 has proven to be a good approach, but it may be possible to improve on it. An alternative model may be found in the field of time series analysis, since the 'condition' of each patient can be seen as a hidden state that evolves through time, and a patients chances of survival depend on his/her condition. Further, Chapter 2 describes a number of existing, alternative approaches to survival analysis, some of which may still be improved by a Bayesian approach.

The distributions that are used in Chapters 3 and 4 are all single Gaussians. Although such distributions have computationally very desirable qualities, other distributions may contain interesting aspects as well, and may be more suitable for some tasks. The model presented in Chapter 4 features purely continuous states. In further research this model may be extended to allow for discrete states. Chapters 3 and 4 may be further combined to incorporate task clustering in the dynamic hierarchical model. In this case, each cluster of models would have the form of the graphical model described in Chapter 3: a set of parallel dynamic linear models that are individually connected through time, and all to a common, average DLM.

Further developments concerning the concept of model clustering may well be found in its applications. This thesis demonstrated its use for bootstrapping methods and multitask learning. Other options are the clustering of samples from Bayesian posterior distributions, or detection of similarities within an ensemble of models with different architectures.

Chapter 2

Improving Cox survival analysis with a neural-Bayesian approach

Abstract. In this chapter we show that traditional Cox survival analysis can be improved upon when supplemented with sensible priors and analyzed within a neural Bayesian framework. We demonstrate that the Bayesian method gives more reliable predictions, in particular for relatively small data sets. The obtained posterior (the probability distribution of network parameters given the data) which in itself is intractable, can be made accessible by several approximations. We review approximations by Hybrid Markov Chain Monte Carlo sampling, a variational method and the Laplace approximation. We argue that although each Bayesian approach circumvents the shortcomings of the original Cox analysis, and therefore yields better predictive results, in practice the use of variational methods or Laplace is preferable. Since Cox survival analysis is infamous for its poor results with (too) many inputs, we use the Bayesian posterior to estimate p-values on the inputs and to formulate an algorithm for backward elimination. We show that after removal of irrelevant inputs Bayesian methods still achieve significantly better results than classical Cox.

Adapted from: B.J. Bakker, T. Heskes, J. Neijt and B. Kappen. Improving Cox survival analysis with a neural-Bayesian approach, *Statistics in Medicine*, In Press.

2.1 Introduction

The purpose of survival analysis for statistics in medicine is to estimate a patient's chances of survival as a function of time, given the available medical information at t_0, the time the patient is admitted to the study. A well-known way to conduct such an analysis is Cox's proportional hazards method [25]. In this method the hazard function $h(t; \mathbf{x})$, which estimates the probability density of death occurring at time t (given that the patient has survived up to that time), is a product of two independent parts. The first part is the proportional hazard, $h(\mathbf{x}) = \exp(\mathbf{w}^T\mathbf{x})$, which depends on patient information $\mathbf{x}$ only, the second part is a time-dependent baseline hazard $h_0(t)$. The most obvious weakness of this model is its vulnerability to overfitting on the training set, often resulting in poor predictions for future patients.

In the medical statistical community a considerable interest exists in the application of neural networks in survival analysis [9, 27, 38, 43, 46, 55, 81, 70]. Although many authors use the neural network machinery to model *non*-proportional hazards, the majority of these models is still based on the original Cox method.

The weaknesses of the standard approach have been acknowledged. Solutions exist in the form of weight decay on the model parameters [9] and implementation of penalty terms [55]. However, an integral solution to the shortcomings of the model based on a solid theoretical background still does not exist.

Another important question in survival analysis is how to implement the effect of time in the model. Options are to view time as an input [9, 27] or to see survival analysis as a series of classification problems, where for each time interval the patients are divided into survivors and non-survivors [55, 70, 81].

In this thesis we hold on to the proportional hazards model since Cox analysis is still the standard in statistical medicine. Furthermore, as we will show in Section 2.7.2, a more complex model does not improve performance. We implement a discretized version of this model in the form of a multi-layered perceptron with one hidden unit and exponential transfer functions, as will be shown in Section 2.4. Each output of the network corresponds to an evaluation of the survivor function at a discrete point in time. The possible adverse effects of this discretization are researched in Section 2.7.3.

Our neural interpretation suggests a Bayesian analysis to overcome the weaknesses of the standard approach. In Section 2.5 sensible priors are introduced which, in combination with the available data, lead to a posterior distribution on the model parameters. This posterior is intractable, but with sampling techniques such as Hybrid Markov Chain Monte Carlo (HMCMC) sampling (see e.g. [65]) we can sample from this posterior to obtain an ensemble of neural networks.

A disadvantage of (HMCMC) sampling is that the number of samples that

need to be drawn to describe the posterior properly grows strongly with the number of network parameters. This not only takes a lot of computation time, it also introduces approximation errors. Furthermore, it is difficult to determine when enough samples have been drawn. As an alternative we propose a form of 'ensemble learning'. This term was coined by Hinton and van Camp [44] and has been applied to multi-layered perceptrons and Radial Basis Function networks in [5, 6]. We approximate the posterior by minimizing the Kullback-Leibler (KL) divergence between the exact posterior given by Bayes' formula and an approximating analytical distribution, varying only the parameters of the latter. We also implement a simpler version of this procedure, where instead of minimizing a KL-divergence we make a Laplace approximation of the posterior. We compare all approaches by estimating the predictive qualities of the resulting posterior distributions. The results are summarized and compared with other (neural) approaches in Sections 2.10 and 2.11.

In practice, medical experts do not work with probability distributions over model parameters directly: they rather use the concept of p-values, for example to express the relevance of patient characteristics. The Bayesian posterior distribution provides a direct way to calculate p-values for the inputs to the network (i.e. patient information). Another problem present in most survival analysis projects is the tendency to include many sources of medical data: rather than missing a possibly significant explanatory variable, medical experts generally prefer to consider a wide variety of possible inputs in the survival model. When the relevance of some of these variables cannot be demonstrated with the model at hand, it is desirable to eliminate these 'irrelevant' inputs from the model, leaving only the inputs with a clear effect on the survival of the patient (see e.g. [3, 45, 83]). In Section 2.8 we propose a Bayesian algorithm to select the relevant inputs of the model and use it to remove the irrelevant variables from the models. After this improvement of the survival model, both for the discretized Cox model and the Bayesian approach, we make our final comparison between the various approaches.

The proposed methods are tested on a medical database, which is described further in the next section.

2.2 Survival analysis on ovarian cancer patients

2.2.1 Survival analysis

In this chapter we study the survival of ovarian cancer patients. Each patient is characterized by a selection of medical data, described in more detail later in this section. Monthly observations of the patients provide information on either the time of death (in months after the initial diagnosis) or survival. The direct task of the methods described in this chapter is to estimate the probability of survival for any patient and for any month, based on the medical

data that is available at the onset of the study. The models providing such estimations will be used to evaluate the (observable) effect of each of the covariates (medical data) on the survival of a patient.

2.2.2 Patients

A database was constructed including 1023 patients from 4 studies, two studies from The Netherlands Joint Study Group for Ovarian Cancer and two from the Gynaecological Cancer Cooperative Group of the European Organisation for Research and Treatment of Cancer (EORTC). The Dutch studies were initiated in 1979 and 1981, respectively. The first study compared a combination of hexamethylmelamine, cyclophosphamide, methotrexate, and 5-fluorouracil (Hexa-CAF) with cyclophosphamide and hexamethylmelamine alternating with doxorubicin and a 5-day course of cisplatin (CHAP-5) in 186 patients with advanced epithelial ovarian carcinoma. In the study initiated in 1981, 191 eligible patients were enrolled and treated with either CHAP-5 or cyclophosphamide and cisplatin (CP), both administered intravenously on a single day at 3-week intervals. Protocol entry criteria, pretreatment staging, histology grading, the randomization procedure, assessment and definitions of tumor response, evaluation and statistical methods were the same in both studies.

The EORTC studies compared CHAP-5 and a combination of cyclophosphamide, hexamethylmelamine alternating with doxorubicin and carboplatin, CHAC-1 (EORTC study no. 55836, 1984), or addressed the question about the efficacy of intervention surgery for patients treated with CP (EORTC study no. 55865, 1987).

We excluded the 94 patients treated with Hexa-CAF, since their medical profile differed largely from the other patients. Of the remaining 929 patients, 246 were censored (i.e. their time of death is not (yet) known). The median observation time of the 929 patients is 653 days.

2.2.3 Missing Data

To handle missing values we distinguished categorical and continuous prognostic variables. For categorical variables we added an additional category representing patients with missing data. For continuous variables we took the average value for that characteristic. Estimating missing values based on the joint probability of missing and known variables did not alter the outcome.

2.2.4 Transformation of data

In the Dutch studies the performance status was registered according to the Karnofsky scale. We reclassified them according to the scale used by the

Zubrod-ECOG-WHO. Patients with a Karnofsky rating of 100% were classified as ECOG 0, those with 90 or 80% as 1, those with 70 or 60% as 2 and patients with a Karnofsky rating less than 60% were classified as 3. When the Broder's grading system was used, we registered a Broder's grade 1 as a well differentiated tumor, Broder's grade 2 as moderately differentiated and Broder's grades 3 and 4 as poorly differentiated. The dose-intensity of each drug was expressed as mg/m^2/wk administered during the first treatment cycle.

2.2.5 Variables

Thirty one variables have been processed by the network: tumor size before and after initial surgery, cell type (serous, endometrioid, mucinous, clear cell, undifferentiated, unclassified, other), grade, weight (kg), length, body surface (m^2), age, treatment, carboplatin dose in cycle 1 divided by the area under the curve (AUC), dose in cycle 1 of doxorubicin, cyclophosphamide, hexamethylmelamine, cisplatin, carboplatin, dose intensity in cycle 1 (mg/m^2/wk), number of sites before and after surgery, the presence of ascites, FIGO stage, performance status according to WHO criteria, hemoglobin (mmol/L), leucocytes (10^9/L), lowest leucocytes (10^9/L), renal clearance (mL/min), number of thrombocytes (10^6/L), lowest number of thrombocytes (10^6/L), serum creatinine (mmol/L), bilirubin (μmol/L).

In order to compare these factors with each other, they were rescaled. For each factor the mean and standard deviation were determined and the mean subtracted from the input values. The result was divided by the standard deviation in such a way that each of the characteristics had a mean equal to zero and a standard deviation equal to one.

2.3 Cox survival analysis

Given the hazard function $h(t;\mathbf{x}) = \exp(\mathbf{w}^T\mathbf{x})h_0(t)$ the survivor function $F(t;\mathbf{x})$ indicating the probability to survive up to time t can be formulated as

$$F(t;\mathbf{x}) = \exp\left[-\int_0^t dt'\ h(t';\mathbf{x})\right] . \tag{2.1}$$

The probability density $f(t;\mathbf{x})$ for a patient to expire at time t is then given by

$$f(t;\mathbf{x}) = -\frac{\partial F(t;\mathbf{x})}{\partial t} = h(t;\mathbf{x})F(t;\mathbf{x}) . \tag{2.2}$$

The likelihood function $P(D|\mathbf{w}, h_0)$, expressing the probability to observe the data in database D given the model parameters $\mathbf{w}$ and the specific choice for the baseline hazard h_0, then immediately follows as

$$P(D|\mathbf{w}, h_0) = \prod_{\nu \in \text{uncensored}} f(t^\nu; \mathbf{x}^\nu) \prod_{\mu \in \text{censored}} F(t^\mu; \mathbf{x}^\mu) \,. \tag{2.3}$$

The first product is over the patients of whom the time of death is known. An element in the second product specifies the estimated probability of censored patient μ to be alive at time t^μ, the time this patient was taken out of the study. Since in this case the time of death is not known this is the strongest prediction that can be verified.

In classical Cox analysis the model parameters are estimated through optimization of the likelihood $P(D|\mathbf{w}, h_0)$ with respect to $\mathbf{w}$ and h_0. An advantage of Cox analysis is that the optimal parameters of the proportional and the time-dependent hazard can be found sequentially. The optimal choice for the parameters $\mathbf{w}$ ($\mathbf{w}^{\text{ML}}$) depends only on the ordering of the times of death of the patients (see [25]); all other time-dependent information is modeled in the function $h_0(t)$.

Disadvantages of this approach are the tendency of the hazard to become highly non-smooth and the danger of strongly over-fitting the data. In this thesis we will eliminate these disadvantages of Cox analysis through the introduction of sensible priors (Section 2.5). We will show (Section 2.7 and further) that by doing so we obtain a stable, reliable model, which still preserves the elegance and simplicity of the original method.

2.4 A discretized model

The first step in our approach of survival analysis is to define a framework of parameters that specifies the hazard function. The parameters for the proportional hazard, $h(\mathbf{x}) = \exp(\mathbf{w}^T\mathbf{x})$, have already been defined. However, the baseline hazard h_0 in classical Cox analysis is a free, continuous function of time. To find the optimal choice for this function one must either choose a specific functional form for the baseline hazard and optimize the corresponding function parameters, or discretize $h_0(t)$ over time. Since the latter option provides more freedom for the form of $h_0(t)$ we will use a discretized version of survival analysis, estimating values for $h_0(t_k)$ for specific discrete points in time. Between two points in time, say t_{k-1} and t_k, we take the hazard function $h(t; \mathbf{x})$ to be constant, i.e.

$$h(t; \mathbf{x}) = h(t_k; \mathbf{x}), \qquad t_{k-1} < t \leq t_k, \tag{2.4}$$

where we take all time intervals (t_{k-1}, t_k) to have the same size, determined by the length of the study and the desired number of discrete model outputs.

In this way, although the baseline hazard is not inferred directly from the data for each of the infinitely many points in time, it is not restricted to a specific functional form either. In Section 2.7.3 we will show that for the right number of discrete outputs the model will not suffer from the inaccuracy brought into it by discretization.

After division of the time span in discrete intervals of length Δt the survivor function reads:

$$F(t;\mathbf{x}) = \exp\left[-\sum_{i=1}^{k-1} \Delta t\ h(t_i,\mathbf{x}) - (t-t_{k-1})h(t_k,\mathbf{x})\right], t_{k-1} < t \leq t_k, \quad (2.5)$$

changing Equation (2.3) to

$$P(D|\mathbf{w},h_0) = \prod_{\nu\in\text{uncensored}} h(t_k^\nu;\mathbf{x}^\nu)F(t^\nu;\mathbf{x}^\nu) \prod_{\mu\in\text{censored}} F(t^\mu;\mathbf{x}^\mu), \quad (2.6)$$

where

$$t_{k-1}^\nu < t^\nu \leq t_k^\nu, t_{k-1}^\mu < t^\mu \leq t_k^\mu. \quad (2.7)$$

Optimization of the discrete likelihood function proceeds in much the same way as in the classical Cox method, resulting in what we will call the maximum likelihood (ML) Cox solution.

The discrete survivor function can be represented by a multi-layered perceptron with exponential transfer functions, as can be seen in Figure 2.1 (concentrate for the moment only on the solid lines). The input $\mathbf{x}$ consists of the elements of patient information, such as the type of medication administered or the presence of specific symptoms. The weights $\mathbf{w}$ in the first layer (input to hidden) correspond to the parameters $\mathbf{w}$ in the proportional hazard function $\exp(\mathbf{w}^T\mathbf{x})$ which is the output of the hidden unit. When the weights $\mathbf{s}$ in the second layer (hidden to output) are equated to

$$s_i = -\int_0^{t_i} dt'\ h_0(t'), \quad (2.8)$$

i.e. minus the integral up to time t_i of the base-line hazard, the output of the network at neuron i reads

$$F_i(\mathbf{x}) = \exp\left[-\int_0^{t_i} dt'\ h_0(t')\exp(\mathbf{w}^T\mathbf{x})\right], \quad (2.9)$$

which, as in Equation (2.1), equals the probability for a patient with characteristics $\mathbf{x}$ to survive up to time t_i. We further define s_i as

$$s_i = -\sum_{j=1}^{i} \exp(v_j)\,, \quad (2.10)$$

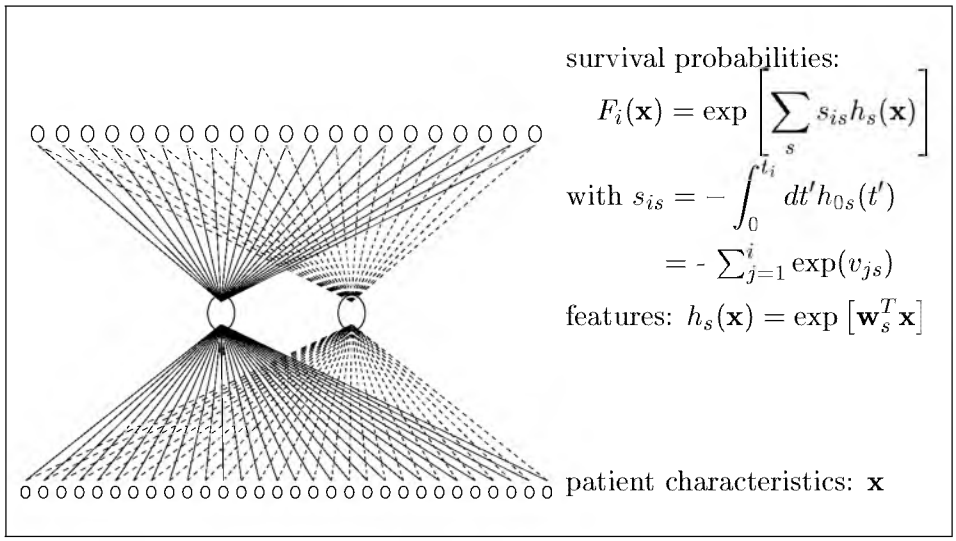


Figure 2.1: Neural interpretation of survival analysis.

where $(\Delta t)^{-1} \exp(v_j)$ can be seen as a discrete version of the baseline hazard at time t_j.

Given this network structure, it is easy to extend the model beyond proportional hazards. The addition of more hidden units to the second layer (dashed lines) may well provide the possibility to model complex input-output relations that cannot be found in the proportional hazard approach.

2.5 Bayesian inference

A discretized version of the Cox proportional hazards method has been defined in Sections 2.3 and 2.4. In this section we describe a method to find the optimal choice for the model parameters without overfitting on the training data. We propose a Bayesian approach in which a probability distribution over all possible values of the model parameters will be constructed. This distribution will not only depend on the (medical) data, but also on prior knowledge about the nature of the problem. This prior knowledge is expressed in probability distributions of the model parameters, which are called *priors*. Using Bayes' formula, the priors and the data likelihood can be combined in the posterior distribution that describes the probability of any choice for the model parameters $\mathbf{w}$ and $\mathbf{v}$.

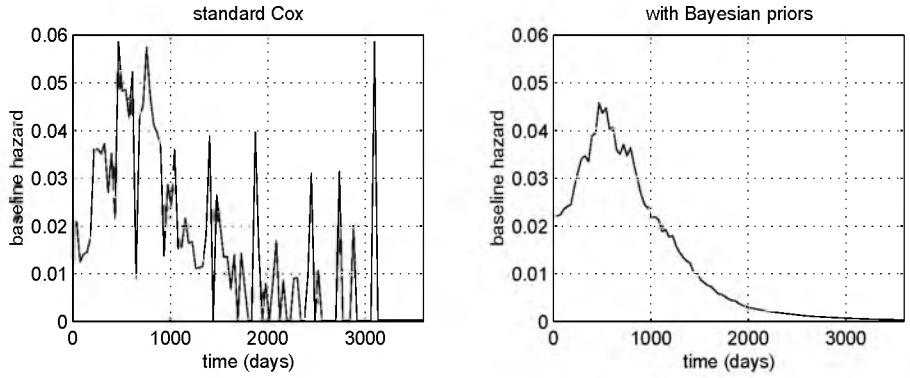


Figure 2.2: The baseline hazard $h_0(t)$ as obtained in classical Cox analysis (left), and after imposing a Bayesian prior on the parameters $\mathbf{v}$ (right). In both cases h_0 is optimized on a training set of 600 patterns, chosen randomly from our database. The effect of the prior is a considerable smoothing of the hazard function.

The first prior

$$P(\mathbf{v}|\gamma) \propto \exp\left[-\frac{\gamma}{2}\sum_{ij} g(|i-j|)[v_i - v_j]^2\right] \propto \exp\left[-\frac{\gamma}{2}\mathbf{v}^T\Gamma\mathbf{v}\right], \qquad (2.11)$$

where $\Gamma_{ij} = -2g(|i-j|), \Gamma_{ii} = 2\sum_{j\neq i} g(|i-j|)$ and we choose $g(z) = \mathrm{e}^{-z^2/\beta}$, prevents the hazard from becoming too sharp as a function of time. In the classical Cox approach the hazard over a specific time interval is directly proportional to the number of casualties in that interval. Although this may give a good representation of the part of the data the model is trained on, it provides poor generalization and yields highly non-smooth functions which are not intuitively plausible. Since $P(\mathbf{v}|\gamma)$ assigns the highest likelihood to a hazard function which is constant in time, it smoothes out the hazard function and introduces a preference for survivor functions which decay exponentially.

The effect of this prior is visualized in Figure 2.2. The hazard function in the ML Cox approach is a jagged function, due to the limited information in the database. After imposing the prior $P(\mathbf{v}|\gamma)$ this function becomes much more smooth. This smoother function is not only more plausible *a priori* but, as will be shown in Section 2.7, it also has a large positive effect on the predictive qualities of the model.

The second prior

$$P(\mathbf{w}|\lambda) \propto \exp\left[-\frac{\lambda}{2}\mathbf{w}^T\Lambda\mathbf{w}\right], \quad \text{where} \quad \Lambda = \frac{1}{\#\ patients}\sum_{\mu}\mathbf{x}^{\mu}\mathbf{x}^{\mu T}, \qquad (2.12)$$

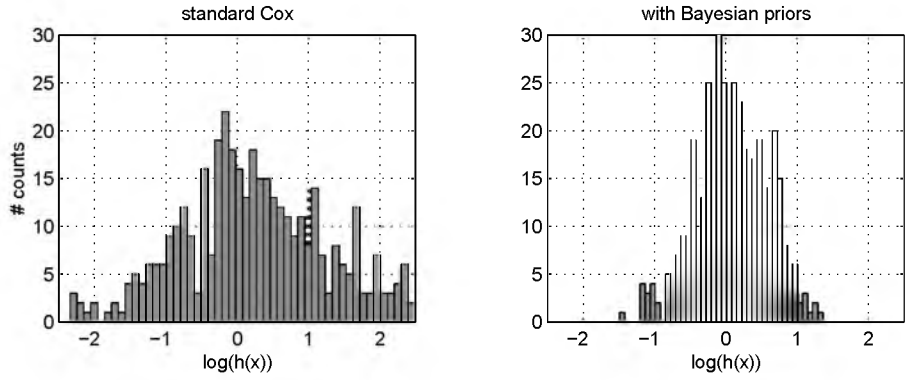


Figure 2.3: Histograms of the (log of the) proportional hazard $h(\mathbf{x})$ as obtained in classical Cox analysis (left), and after imposing a Bayesian prior on the parameters $\mathbf{w}$ (right). In both cases the parameters $\mathbf{w}$ of the proportional hazard are optimized on a training set of 600 patterns, chosen randomly from our database. The effect of the prior is a considerable 'shrinking' of the proportional hazard.

prevents large activities of hidden units (high values for the proportional hazard), i.e. prefers small weights. This prior corresponds to a ridge-type estimator, as discussed in [33]. Incorporation of the covariance matrix Λ makes this preference independent of a (linear) scaling of the inputs $\mathbf{x}$. The effect of imposing this prior on the parameters $\mathbf{w}$ is shown in Figure 2.3. It can be seen that in the unrestrained case (left panel) it occurs quite often that the proportional hazard of one patient is up to ten times larger than it is for another patient, indicating that the model may be overfitting on the training data. After imposing a Bayesian prior on the model these differences become much more reasonable.

The hyperparameters γ and λ express the confidence one has in the knowledge expressed in the two prior distributions. Since we do not want to specify the exact values of γ and λ, we introduce gamma distributions (defined as $P(\lambda|\sigma,\tau) = \frac{\tau^\sigma}{\Gamma(\sigma)}\lambda^{\sigma-1}\exp(-\tau\lambda))$ $P(\lambda)$ and $P(\gamma)$ for the hyperparameters. The expectation value and the variance of λ are calculated as $\frac{\sigma}{\tau}$ and $\frac{\sigma}{\tau^2}$, respectively. Choosing values for these two ratios enables us to specify our believes about the model in more detail. We use the visualization of the effect of our first prior (Figure 2.2) to judge which γ yields acceptable (smooth) hazard functions. In estimating the expected value for λ we can use Figure 2.3, or we can consider what patient to patient variation in diagnosis may still be acceptable. Using this decision, λ can be set to assign the desired *a priori*

probability of the parameters $\mathbf{w}$. The variance will be kept relatively high to obtain very wide hyperpriors.

The posterior distribution of the parameters $\mathbf{w}$ and $\mathbf{v}$ and hyperparameters λ and γ given the data follows from Bayes' formula:

$$P(\mathbf{w}, \mathbf{v}, \lambda, \gamma | D) = \frac{P(D|\mathbf{w}, \mathbf{v}) P(\mathbf{w}|\lambda) P(\mathbf{v}|\gamma) P(\lambda) P(\gamma)}{P(D)} , \tag{2.13}$$

with $P(D|\mathbf{w}, \mathbf{v})$ the likelihood as in (2.6) and $P(D)$ an irrelevant normalizing constant. The probability density of any conceivable choice for the model parameters $\mathbf{w}$ and $\mathbf{v}$ is given by the posterior $P(\mathbf{w}, \mathbf{v}|D)$, which follows by integrating out the hyperparameters λ and γ.

2.6 Approximation of the posterior

2.6.1 Three methods

In theory the expression for the posterior probability distribution of the model parameters is all that is needed to make predictions for new patients. First however, the posterior distribution of model parameters and hyperparameters (Equation (2.13)) needs to be transformed into a distribution of the model parameters alone. To this end we must perform the integral over the hyperparameters λ and γ. Since this cannot be done analytically, we have to make an approximation of the true posterior.

Fortunately, an ample arsenal of methods to approximate $P(\mathbf{w}, \mathbf{v}|D)$ is available. These methods have become popular tools in the neural networks community. In this section, we will apply and evaluate three such methods: Hybrid Markov Chain Monte Carlo Sampling (HMCMC), a variational approach and the Laplace approximation.

2.6.2 Hybrid Markov Chain Monte Carlo Sampling

Since the posterior $P(\mathbf{w}, \mathbf{v}, \lambda, \gamma|D)$ is not a simple analytic function of the model parameters, it is hard to draw samples from it. Therefore, we will use both Gibbs sampling and HMCMC to make it tractable. In Gibbs sampling, one starts by taking random values for the hyperparameters, say $\tilde{\lambda}$ and $\tilde{\gamma}$. Next, $P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma}, D)$ is obtained from

$$\begin{aligned} P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma}, D) &= \frac{P(D|\mathbf{w}, \mathbf{v}, \tilde{\lambda}, \tilde{\gamma}) P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma})}{P(D|\tilde{\lambda}, \tilde{\gamma})} \\ &\propto P(D|\mathbf{w}, \mathbf{v}) P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma}) , \end{aligned} \tag{2.14}$$

the product of the likelihood and the priors, with fixed values for the hyper-parameters. Now, in turn, samples are drawn from $P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma}, D)$ yielding $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$, and from $P(\lambda, \gamma|\tilde{\mathbf{w}}, \tilde{\mathbf{v}})$,

$$\begin{aligned} P(\lambda, \gamma|\tilde{\mathbf{w}}, \tilde{\mathbf{v}}) &= \frac{P(\tilde{\mathbf{w}}, \tilde{\mathbf{v}}|\lambda, \gamma)P(\lambda, \gamma)}{\int d\lambda d\gamma P(\tilde{\mathbf{w}}, \tilde{\mathbf{v}}|\lambda, \gamma)P(\lambda, \gamma)} \\ &\propto P(\tilde{\mathbf{w}}|\lambda)P(\tilde{\mathbf{v}}|\gamma)P(\lambda)P(\gamma)\,, \end{aligned} \tag{2.15}$$

which is itself a gamma distribution. Since $P(\mathbf{w}, \mathbf{v}|\tilde{\lambda}, \tilde{\gamma}, D)$ does not have the form of one of the standard distributions (normal, gamma) we use HMCMC to sample from it.

Hybrid Markov Chain Monte Carlo sampling [60] is a well-known method for sampling from complex distributions. The first step in HMCMC is to express $P(\mathbf{q})$, the distribution we wish to sample from, as

$$P(\mathbf{q}) = \exp(-E(\mathbf{q})), \tag{2.16}$$

where in the case of the model discussed in this chapter, $\mathbf{q}$ represents the parameters $\{\mathbf{w}, \mathbf{v}\}$. Further we define the canonical distribution

$$P(\mathbf{p}) = Z^{-1}\exp(-K(\mathbf{p})), \tag{2.17}$$

where

$$K(\mathbf{p}) = \sum_{i=1}^{n} \frac{p_i^2}{2m_i} \tag{2.18}$$

and Z is a normalizing constant, so that each p_i is normal distributed around zero with variance m_i. n is the dimension of $\mathbf{p}$, which is equal to the dimension of $\mathbf{q}$. The joint distribution $P(\mathbf{q}, \mathbf{p})$ is defined as

$$P(\mathbf{q}, \mathbf{p}) = \exp(-E(\mathbf{q}))Z^{-1}\exp(-K(\mathbf{p}))\,. \tag{2.19}$$

Now we sample from the joint distribution $P(\mathbf{q}, \mathbf{p})$. We first take a random choice for $\mathbf{q}$ and draw a sample $\mathbf{p}$ from the normal distribution $K(\mathbf{p})$. Next, this sample is transformed by applying the following dynamics:

$$\frac{dq_i}{d\tau} = \frac{p_i}{m_i}, \tag{2.20}$$

and

$$\frac{dp_i}{d\tau} = -\frac{\partial E}{\partial q_i}, \tag{2.21}$$

where τ serves as a simulated 'time' parameter. It can be shown that under these rules the probability $P(\mathbf{q}, \mathbf{p})$ remains the same. Therefore, the simulation of the evolution of $\mathbf{q}$ and $\mathbf{p}$ in 'time' provides a new and -if a sufficient number

of steps in 'time' have been taken- independent sample $(\mathbf{q}^*, \mathbf{p}^*)$, which has the same probability as the old sample $(\mathbf{q}, \mathbf{p})$. Since $\mathbf{q}$ and $\mathbf{p}$ are independent under the distribution $P(\mathbf{q}, \mathbf{p})$, this also yields a new sample $\mathbf{q}^*$ of $P(\mathbf{q})$. Drawing a new sample from $P(\mathbf{p})$ completes the new sample $(\mathbf{q}, \mathbf{p})$, on which the dynamics (2.21) can be applied again, etc. This way, neglecting $\mathbf{p}$, samples are drawn effectively from $P(\mathbf{q})$.

However, because the applied 'time evolution' proceeds in finite steps, the probability $P(\mathbf{q}, \mathbf{p})$ may not be conserved exactly. Therefore, before accepting a new sample $\mathbf{q}^*$ the ratio of the joint probability of the current sample $\{\mathbf{p}^*, \mathbf{q}^*\}$ and the previous sample $\{\mathbf{p}, \mathbf{q}\}$,

$$R_M = \frac{P(\mathbf{q}^*, \mathbf{p}^*)}{P(\mathbf{q}, \mathbf{p})} , \tag{2.22}$$

must be considered. If $R_M > 1$ the new sample is accepted always, otherwise it is accepted with probability R_M. It can be shown [60] that in this way, in spite of the computational inaccuracy introduced in (2.21), one still samples from $P(\mathbf{q})$. The first few samples may suffer from initialization effects and therefore cannot be trusted to give a good representation of the posterior. Therefore, after sampling we will discard the first 100 samples.

The advantage of this method is that large jumps in '$\mathbf{q}$-space' may be taken (there can be large differences between subsequent samples of $\mathbf{q}$), yielding a series of independent samples, which is not the case in standard Markov chain sampling. Further, if the time steps in (2.21) are chosen small enough very few samples will be rejected.

Adding extra hidden units does not change the format of this sampling procedure. It does however cause the number of parameters, and therefore the number of samples that needs to be drawn, to increase strongly.

2.6.3 Variational approach

Another approximation of $P(\mathbf{w}, \mathbf{v}|D)$ can be made by fitting an analytical distribution to the exact posterior. An advantage of this approach is that we obtain a simple expression for the posterior. Following Barber and Bishop [5] we approximate the joint posterior distribution of model parameters and hyperparameters $P(\mathbf{w}, \mathbf{v}, \lambda, \gamma|D)$ by a factorized distribution of the form

$$P^*(\mathbf{w}, \mathbf{v}, \lambda, \gamma) = Q(\mathbf{w}, \mathbf{v}) R(\lambda) S(\gamma) , \tag{2.23}$$

with $Q(\mathbf{w}, \mathbf{v}) = \mathcal{N}(\hat{\mathbf{w}}, \hat{\mathbf{v}}, C)$, a Gaussian distribution with mean $\{\hat{\mathbf{w}}, \hat{\mathbf{v}}\}$ and covariance matrix C, and $R(\lambda)$ and $S(\gamma)$ for the moment unspecified. We assume that there is no interaction between $\mathbf{w}$ and $\mathbf{v}$, so C can be written as

$$C = \begin{pmatrix} C_{ww} & \emptyset \\ \emptyset & C_{vv} \end{pmatrix} , \tag{2.24}$$

which yields a significant reduction in the number of free parameters. To reduce the number of free parameters in the covariance matrix further, instead of using the full covariance matrix C_{vv} we take a constrained matrix $\tilde{C}_{vv}$ specified by

$$\left[\tilde{C}_{vv}^{-1}\right]_{ij} = k_i \ \exp(-\Delta_{ij}) \ k_j \tag{2.25}$$

where $\mathbf{k}$ is an unconstrained vector and

$$\Delta_{ij} = |i - j|, \tag{2.26}$$

which we found to be a good approximation of the real covariance matrix. The strongly reduced number of free parameters in $\tilde{C}_{vv}$ (now equal to the number of discrete model outputs) makes it possible to use a fine discretization in time, i.e. a large number of model outputs. The covariance matrix C_{ww} is left unconstrained.

As a distance measure between $P(\mathbf{w}, \mathbf{v}, \lambda, \gamma|D)$ and $P^*(\mathbf{w}, \mathbf{v}, \lambda, \gamma)$ we use the Kullback-Leibler divergence

$$\begin{aligned} \mathrm{KL}[Q,R,S] &= \int d\mathbf{w}d\mathbf{v}d\lambda d\gamma \ Q(\mathbf{w},\mathbf{v})R(\lambda)S(\gamma) \log\left[\frac{Q(\mathbf{w},\mathbf{v})R(\lambda)S(\gamma)}{P(\mathbf{w},\mathbf{v},\lambda,\gamma|D)}\right] \\ &\equiv \langle \log\left[Q(\mathbf{w},\mathbf{v})R(\lambda)S(\gamma)\right]\rangle_{Q,R,S} - \\ &\quad \langle \log\left[P(D|\mathbf{w},\mathbf{v})P(\mathbf{w}|\lambda)P(\mathbf{v}|\gamma)P(\lambda)P(\gamma)\right]\rangle_{Q,R,S} \ , \end{aligned} \tag{2.27}$$

where in the third line we substituted Equation (2.13). The goal is to find the normal distribution $Q(\mathbf{w}, \mathbf{v})$ and additional distributions $R(\lambda)$ and $S(\gamma)$ that minimize this distance.

The Kullback-Leibler divergence depends both on the choice of parameters $\{\hat{\mathbf{w}}, \hat{\mathbf{v}}, C\}$ and on the distributions $R(\lambda)$ and $S(\gamma)$. Let us first suppose that $R(\lambda)$ and $S(\gamma)$ are given. The terms in (2.27) depending on $Q(\mathbf{w}, \mathbf{v})$ and thus on the variational parameters $\{\hat{\mathbf{w}}, \hat{\mathbf{v}}, C\}$ are

$$\begin{aligned} \mathrm{KL}[Q] &= \langle \log Q(\mathbf{w},\mathbf{v})\rangle_Q - \langle \log P(D|\mathbf{w},\mathbf{v})\rangle_Q - \\ &\quad \langle \log P(\mathbf{w}|\lambda)\rangle_{Q,R} - \langle \log P(\mathbf{v}|\gamma)\rangle_{Q,S} \ . \end{aligned} \tag{2.28}$$

Except for the second term involving the data likelihood, all terms are straightforward. Neglecting irrelevant constants we have

$$\langle \log Q(\mathbf{w},\mathbf{v})\rangle_Q = -\frac{1}{2}\log|C| \ \text{ and} \tag{2.29}$$

$$\langle \log P(\mathbf{w}|\lambda)\rangle_{Q,R} = -\frac{\bar{\lambda}}{2}\left[\hat{\mathbf{w}}^T\Lambda\hat{\mathbf{w}} + \mathrm{Tr}\ (C_{ww}\Lambda)\right] \ , \tag{2.30}$$

with $\bar{\lambda} \equiv \langle\lambda\rangle_R$, and a similar expression for $\langle \log P(\mathbf{v}|\gamma)\rangle_{Q,S}$. The likelihood term $\langle \log P(D|\mathbf{w},\mathbf{v})\rangle_Q$ can again be decomposed into two terms (see (2.6)):

a term involving only uncensored patients and a term to which all patients contribute. Both contributions can be computed analytically. The uncensored patients ν yield

$$\left\langle \log \left[\mathrm{e}^{v_k} \mathrm{e}^{\mathbf{w}^T \mathbf{x}^\nu} \right] \right\rangle_Q = \hat{v}_k + \hat{\mathbf{w}}^T \mathbf{x}^\nu, t_{k-1} < t^\nu \leq t_k \,, \tag{2.31}$$

and all patients μ contribute terms of the form

$$-\left\langle \mathrm{e}^{v_i} \mathrm{e}^{\mathbf{w}^T \mathbf{x}^\mu} \right\rangle_Q = -\exp\left[\hat{\mathbf{w}}^T \mathbf{x}^\mu + \hat{v}_i + \frac{1}{2} \mathbf{x}^{\mu T} C_{ww} \mathbf{x}^\mu + \frac{1}{2} \tilde{C}_{v_i v_i} \right] \,, \tag{2.32}$$

where we have used the equality

$$\int d\mathbf{y} \; P(\mathbf{y}) \mathrm{e}^{\mathbf{y}^T \mathbf{z}} = \exp\left[\mathbf{m}^T \mathbf{z} + \frac{1}{2} \mathbf{z}^T \Sigma \, \mathbf{z} \right] \quad \text{with} \quad P(\mathbf{y}) = \mathcal{N}(\mathbf{m}, \Sigma) \,, \tag{2.33}$$

in which we substitute $\{\mathbf{w}, \mathbf{v}\}$ for $\mathbf{y}$ and the vector $\{\mathbf{x}, [\ldots, 0, 0, 1, 0, 0, \ldots]\}$, with 1 at the position of i, for $\mathbf{z}$.

Summarizing, given the values for $\bar{\lambda}$ and $\bar{\gamma}$, the variational parameters $\{\hat{\mathbf{w}}, \hat{\mathbf{v}}, C\}$ of $Q(\mathbf{w}, \mathbf{v})$ can be found by minimizing the error function KL$[Q]$, for example using a conjugate gradient method.

Now suppose that $Q(\mathbf{w}, \mathbf{v})$ is known and we would like to optimize for $R(\lambda)$. The terms in (2.27) which depend on $R(\lambda)$ are

$$\mathrm{KL}[R] = \langle \log R(\lambda) \rangle_R - \langle \log P(\lambda) \rangle_R - \langle \log P(\mathbf{w}|\lambda) \rangle_{Q,R} \,. \tag{2.34}$$

It is easy to show that, with a Gaussian prior $P(\mathbf{w}|\lambda)$ and a gamma distribution for $P(\lambda)$, the optimal $R(\lambda)$ is also gamma distributed (see e.g. [5] for details). The procedure for $S(\gamma)$ is completely equivalent.

The approximate posterior $P^*(\mathbf{w}, \mathbf{v}, \lambda, \gamma)$ can now be found by iterating the following two steps.

- Minimize KL$[R, S]$ to find $R(\lambda)$ and $S(\gamma)$ and calculate the expectation values $\bar{\lambda}$ and $\bar{\gamma}$ given these distributions.
- Substitute $\bar{\lambda}$ and $\bar{\gamma}$ in KL$[Q]$ and minimize this expression.

Since both steps decrease the value of the total divergence KL$[Q, R, S]$, this iterative procedure will converge to an (at least locally) optimal distribution $P^*(\mathbf{w}, \mathbf{v}, \lambda, \gamma)$. This is actually similar to the Gibbs sampling procedure described in Section 2.6.1, where now instead of sampling from $P(\lambda, \gamma|\bar{\mathbf{w}}, \bar{\mathbf{v}}, D)$ and $P(\mathbf{w}, \mathbf{v}|\bar{\lambda}, \bar{\gamma}, D)$, we obtain analytical expressions for both distributions (which, for $P(\lambda, \gamma|\mathbf{w}, \mathbf{v}, D)$ is merely a matter of writing down the correct expressions) and calculate the expectation values for $\{\lambda, \gamma\}$ and $\{\mathbf{w}, \mathbf{v}\}$ respectively.

Note that, due to the exponential transfer functions and our particular choice of parameterization and corresponding priors, all integrals in KL$[Q, R, S]$ can be done analytically. Therefore, although the specification of the terms of the Kullback-Leibler divergence may look rather complicated, the actual approximation consists of no more than a simple minimization process. This makes the application of the variational approach to survival analysis especially attractive, in contrast with applications to neural networks with sigmoidal transfer functions [5], where numerical evaluations are unavoidable. However, with more than one hidden unit either the term (2.31) involving only uncensored patients, or, with a different parameterization and choice of priors, the contributions (2.32) involving all patients would require numerical integration scaling with the number of added hidden units.

2.6.4 Laplace approximation

The procedure of Section 2.6.3 can be simplified by replacing the minimization of the Kullback-Leibler divergence by a Laplace approximation. We again take a normal distribution $\mathcal{N}(\hat{\mathbf{w}}, \hat{\mathbf{v}}, C)$ for $Q(\mathbf{w}, \mathbf{v})$ with

$$\{\hat{\mathbf{w}}, \hat{\mathbf{v}}\} = \underset{\{\mathbf{w},\mathbf{v}\}}{\operatorname{argmax}} P(\mathbf{w}, \mathbf{v}|\lambda, \gamma, D) \tag{2.35}$$

and C^{-1} the Hessian of $-\log(P(\mathbf{w}, \mathbf{v}|\lambda, \gamma, D))$ at $\mathbf{w} = \hat{\mathbf{w}}$ and $\mathbf{v} = \hat{\mathbf{v}}$. We iterate the same two steps as in Section 2.6.3, only instead of minimizing KL$[Q]$, we implement the Laplace approximation (with fixed values $\bar{\lambda}$ and $\bar{\gamma}$ for λ and γ). Note that in this case no restrictions are imposed on C. The other parts of this approximation are equal to the variational procedure described in Section 2.6.3. This method to approximate the posterior corresponds to the 'evidence framework' introduced in [59].

2.7 Predictive qualities: an evaluation

2.7.1 Model comparison

The approximations of the posterior, described in Section 2.6, enable us to demonstrate the improvement that is gained by the introduction of Bayesian priors. Since we have proposed three different methods to approximate the posterior, we will not only compare the Bayesian approach to survival analysis to the ML Cox method, but we will also find out which of the three approximations yields the best predictions.

To test the different types of models in this chapter we employ a database of 929 ovarian cancer patients of whom, apart from their medical information at the time of entrance to the study, either their time of death is known, or the last date that they were observed to be alive. For test purposes this

database is randomly divided into a training set and a test set. To estimate the predictive qualities of the survival model obtained from the approximations of the posterior as defined by the two priors and the likelihood based on the data in the training set (Equation (2.13)) we compare it to the maximum likelihood fit of the Cox model on the same set. As an error measure we take

$$\mathcal{E} = -\log \mathcal{L}(D_v), \tag{2.36}$$

where $\mathcal{L}(D_v)$ is the likelihood of the data in the validation set, estimated either by the maximum likelihood Cox model or the solution obtained by Bayesian methods. The likelihood estimated by the ML solution is calculated simply by inserting $\{\mathbf{w}^{\mathrm{ML}}, \mathbf{v}^{\mathrm{ML}}\}$ for the model parameters $\{\mathbf{w}, \mathbf{v}\}$. In the Bayesian approach we use

$$\mathcal{L}(D_v) = \prod_{\nu \in \text{uncensored}} \langle f(t^\nu; \mathbf{x}^\nu) \rangle \prod_{\mu \in \text{censored}} \langle F(t^\mu; \mathbf{x}^\mu) \rangle \ , \tag{2.37}$$

where $\langle .. \rangle$ denotes the expectation value over the posterior. In the sampling approach this expectation value is calculated through averaging over all samples after initialization (see Section 2.6.2). In both the variational approach and the Laplace approximation ideally we would use the obtained normal distributions to calculate the expectation value analytically. However, due to the specific form of the likelihood this is not possible. Instead we sample from the normal distributions and take the average over the collected samples, just as in the HMCMC approach. Note however that sampling from a normal distribution is much easier and far less time consuming than sampling from the exact posterior distribution.

To get a clear indication of the relative strengths of both methods we also compute the ML solution on the test set, and use (2.36) to calculate a 'minimum error' $\mathcal{E}_{\min}$. The relative error used in our comparisons now reads

$$\mathcal{E}_{\mathrm{rel}} = \frac{\mathcal{E} - \mathcal{E}_{\min}}{\mathcal{E}_{\mathrm{cox}} - \mathcal{E}_{\min}}, \tag{2.38}$$

where $\mathcal{E}_{\mathrm{cox}}$ is the test error obtained from the ML Cox method. So, if one of the Bayesian methods is just as good as the Cox method it would have a relative error of 1.

The results (Figure 2.4c) show that the errors in any of the approximations to the Bayesian posterior are significantly ($p \approx 1 \times 10^{-5}$) smaller than the error in the classical Cox approach. A closer look at the difference between the three Bayesian approaches reveals that the error in the variational approach is slightly (but significantly) larger than the error in the sampling approach. The Laplace approximation, which takes about as much computation time as the variational approach, does not perform significantly better or worse.

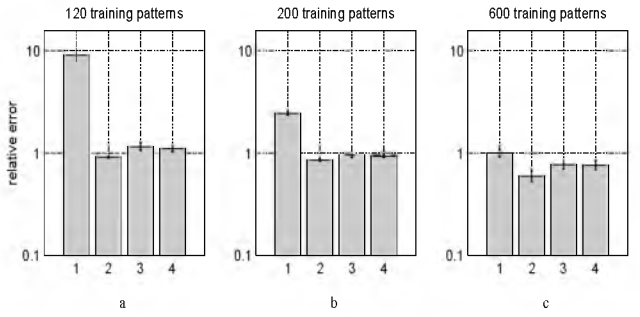


Figure 2.4: Relative error after training on three partitions of the database (120 patients in the left panel, 200 patients in the middle panel and 600 patients in the right panel). In each panel, from left to right the bars represent the error in: the ML Cox method(1), the HMCMC sampling approach(2), the variational approach(3) and the Laplace approximation(4), all with one hidden unit. All differences are significant, except for the one between the variational approach and the Laplace approximation.

The size of the database that we have access to (929 patients) is much larger than is common in survival analysis. Most medical databases in this field consist of 100-200 patients. To show the effect of the size of the available data on the quality of the model we trained the model on smaller parts (120 and 200 patients) of our database, using the same approaches as in the previous section. The results can be seen in Figures 2.4a and b, where for comparison we have used the values for $\mathcal{E}_{\rm cox}$ and $\mathcal{E}_{\rm min}$ obtained on the training set of 600 patterns to scale the errors corresponding to the smaller databases (but on the same validation sets) as well. For lower and lower numbers of training patterns the error in the classical Cox method increases dramatically. The error in the Bayesian approaches also increases slightly, but due to the effect of the sensible priors the Bayesian method is much more stable under decrease of the number of training patterns.

This effect is not very surprising: for the commonly observed database in the field of survival analysis overfitting is a very serious problem due to the small number of patients. Here, enormous improvements can be made by applying Bayesian priors. For larger and larger databases the overfitting problem grows smaller and smaller until finally, in the limit of an infinite number of patients, ML Cox and the Bayesian approach would coincide. Note however that even on a training set of 600 patterns the Bayesian approach yields significantly better results than the ML Cox method.

The comparison made here, with the complete model considering all inputs, may however not be a totally 'fair' one. In the medical statistical community it is well-known that Cox analysis with a full set of inputs strongly suffers from overfitting. A standard procedure to deal with the overfitting problem (and thus the error) in Cox analysis is to reduce the number of inputs to the model. Therefore, in Section 2.8 we will propose a backward elimination procedure, and we will compare the reduced Cox model to the Bayesian approach again in Section 2.9.

2.7.2 Non-proportional hazards

In Section 2.4 we showed that, with some minor adjustments, it is possible to extend the model beyond Cox proportional hazard. To test whether a more complicated model would indeed be more able to fit the data we added one hidden unit (see Figure 2.1). We found that the overfitting problem already present in the ML Cox model with one hidden unit increased strongly after adding a second hidden unit (doubling the number of free model parameters), yielding a relative error which vastly exceeded the error in the simple model. The error in the sampling approach did not change significantly with the addition of an extra hidden unit. In the remainder of this chapter we will concentrate on the version with one hidden unit.

2.7.3 The effect of discretization

To demonstrate the effect of the discretization of the baseline hazard we implemented the ML Cox method and the variational approach on network configurations with numbers of outputs ranging between 5 and 1000 (at this point further discretization was meaningless since no time interval contained more than one patient). The results are shown in Figure 2.5, where the error (2.36) is plotted against the number of outputs. For both the ML Cox method and the Bayesian approach at first an improvement due to finer discretization can be observed, followed by a deterioration due to overfitting on the training data when the number of model (output) parameters grows. For the Bayesian approach the optimal model has a larger number of output parameters since the imposed priors reduce the overfitting problem. From these results it is clear that the model does not suffer from the inaccuracy that is introduced by discretization. In fact, a free, continuous model, corresponding to a very large number of outputs in Figure 2.5, will produce very poor results. Therefore, for the comparison of the models' predictive qualities we use 10 outputs in the ML Cox model and 50 in each of the Bayesian models, corresponding to the observed optima in Figure 2.5.

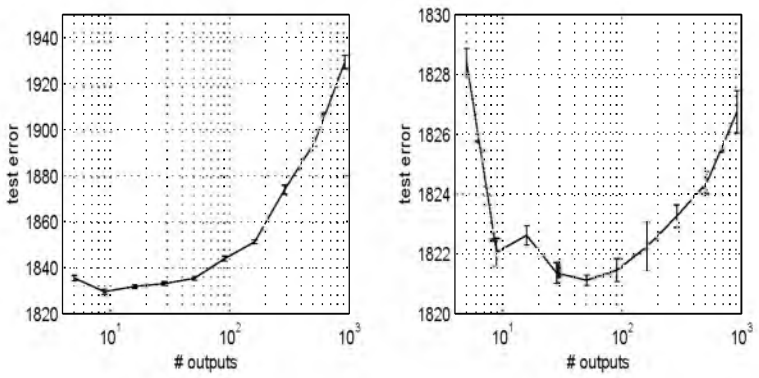


Figure 2.5: Test error as a function of the number of discrete outputs for both the ML Cox method (left panel) and the variational approach (right panel). The errors are the average of the test error obtained from 25 random draws of 600 patients from the database. For each draw we first selected the relevant inputs to the model, as will be described in Sections 2.8 and 2.9, and then varied the number of outputs.

2.8 Bayesian backward elimination

In Section 2.7 we mentioned that the comparison between ML Cox and the Bayesian approach was not completely fair. Therefore, in this section we propose a method to reduce the number of input parameters. After eliminating 'irrelevant' inputs from the model the ML Cox method will be less impaired by overfitting problems. After this improvement, we will be able to make a 'fair' comparison between the Bayesian approach and the ML Cox method (Section 2.9).

In medicine, the calculation of p-values is a well-known tool for statistical analysis. A p-value is defined as follows: consider a null-hypothesis H_0, which is rejected if a certain test statistic T exceeds the critical value T_c. Now, the p-value of a specific measurement t of T is calculated as $P(T > t|H_0)$, the probability of finding a value for T which exceeds t, given that the null-hypothesis is true [61]. The p-value gives an indication of the conflict between the null-hypothesis and the observed data.

To determine the relevance of one of our models input weights, e.g. w_{D}, we define a null-hypothesis that states that the 'true' value of w_{D} is zero. This hypothesis would be rejected if the measured value of w_{D} (the maximum likelihood value $w_{\mathrm{D}}^{\mathrm{ML}}$ for classical Cox, the expectation value $\langle w_{\mathrm{D}} \rangle$ for the Bayesian approach) would exceed some critical value w_c. We derive $P(w_{\mathrm{D}} > w_{\mathrm{D}}^{\mathrm{ML}}|H_0)$ to calculate the p-value for each of the inputs of the model. In this expression,

$P(w_{\mathrm{D}}|H_0)$ is approximated as a normal distribution with mean $\hat{w_{\mathrm{D}}} = 0$, and variance equal to the estimated marginal variance of w_{D} (e.g. $C_{ww,\mathrm{DD}}$ in the variational approach).

Upon constructing the posterior and calculating the p-values of the input weights of the model nearly all weights were deemed irrelevant ($p > 0.05$). This does not necessarily mean that we have created a useless model, it merely states that the function of any input can be taken over by the other inputs if it would be removed. This is a common problem in survival analysis: rather than missing a source of data which might be relevant, medical scientists consider a large variety of possibly interesting sources of medical data. However, since the model is trained on a limited database, not only the desired underlying 'true' relations between the data are modelled, but also random effects and anomalies present in the database. To discern between relevant and irrelevant parameters, c.q. inputs, we will propose and apply a backward elimination procedure.

Backward elimination is a procedure in which one by one all irrelevant model parameters are removed, leaving a completely 'meaningful' model. In each step of the procedure the least relevant model parameter is found and eliminated. The procedure should continue as long as the decrease in the error due to over-fitting outweighs the reduced functionality of the model. Note that backward elimination is a suboptimal heuristic: in principle all possible subsets of parameters should be considered. However, it has been shown to give close to optimal results in many cases [11].

Two elements need to be defined for backward elimination: a criterion to decide which of the model parameters should be eliminated in each step and an indicator to say when to stop removing parameters. The first criterion may depend, for example, on the size of the parameters or on some estimation of the increase in training error. The stopping criterion can be established by cross-validation on a separate validation set, or e.g. by Akaike's Information Criterion. In our case we could use the concept of p-values, removing in each step the parameter with the highest p-value, and stopping when all parameters are relevant ($p < 0.05$). However, many statisticians criticise the concept of p-values [8], [50]; p-values just do not coincide with the characteristic we are interested in: the probability that a certain model parameter is irrelevant or, in a broader sense, the probability of the model given the observed data. In fact, a p-value generally overstates the evidence against the null-hypothesis and thus gives a wrong impression of the relevance of model parameters to researchers without a broad experience working with p-values. We tend to agree with the critics in the field, and therefore use the *Bayes factor* [8, 59] instead. The Bayes factor reads

$$BF = \frac{P(D|H_0)}{P(D|H_1)}, \tag{2.39}$$

where the hypotheses H_0 and H_1 are specified by

$$H_0 \quad : \quad |w_{\mathrm{D}}| < \epsilon, \qquad \epsilon \to 0 \tag{2.40}$$

$$H_1 \quad : \quad |w_{\mathrm{D}}| > \epsilon, \qquad \epsilon \to 0 \tag{2.41}$$

with w_{D} the parameter we consider removing. In other words: the null-hypothesis states that w_{D} is actually zero and can be removed, the alternative hypothesis H_1 that w_{D} is relevant and should stay (the mathematically less critical reader can read $w_{\mathrm{D}} = 0$ for $|w_{\mathrm{D}}| < \epsilon$ and $w_{\mathrm{D}} \neq 0$ for $|w_{\mathrm{D}}| > \epsilon$). The Bayes factor is an expression indicating which of both hypotheses best explains the data D.

Writing the Bayes factor as

$$BF = \lim_{\epsilon \to 0} \frac{P(D||w_{\mathrm{D}}| < \epsilon)}{\int_{|w_{\mathrm{D}}|>\epsilon} dw_{\mathrm{D}}\, d\lambda\, P(D|w_{\mathrm{D}})\; P(w_{\mathrm{D}}|\lambda)\; P(\lambda)}\,, \tag{2.42}$$

we recognize that, since for $\epsilon \to 0$ the integral over $|w_{\mathrm{D}}| > \epsilon$ is equal to the integral over all w_{D}, the lower term in (2.42) yields $P(D)$. Applying Bayes' rule to the upper term, we obtain

$$BF = \lim_{\epsilon \to 0} \frac{P(|w_{\mathrm{D}}| < \epsilon|D)}{P(|w_{\mathrm{D}}| < \epsilon)}\,. \tag{2.43}$$

The Bayes factor (calculated explicitly in Appendix 2.A) provides us with a simple backward elimination algorithm:

- Calculate $\frac{P(D|H_0)}{P(D|H_1)}$ for each candidate w_{D}
- Select the candidate with the highest ratio: if it exceeds one, eliminate the parameter, else stop
- Retrain the reduced network
- Repeat the previous three steps until no irrelevant parameters are left.

Due to the retraining after each step, this can be a time-costly exercise. However, instead of being retrained the posterior can be re-estimated after each parameter removal through the calculation of $P(\mathbf{w}_{\mathrm{R}}|w_{\mathrm{D}} = 0, D)$, where $\mathbf{w}_R$ is the part of $\mathbf{w}$ that remains after deletion of w_{D} (see Appendix 2.B). Note that this expression is just another way of writing 'the distribution with $w_{\mathrm{D}} = 0$ which is closest to the original distribution, $P(\mathbf{w}|H_0)$'. This method has been implemented in [54] and is closely related to a backward elimination algorithm based on the Bayesian evidence framework proposed by MacKay [59].

Backward elimination in the classical Cox approach can be executed using for example a technique that in the neural network community is referred to

as 'Optimal Brain Surgeon' [37]. Here we express the difference between two models with parameters $\mathbf{q}_1 = \{\mathbf{w}_1, \mathbf{v}_1\}$ and $\mathbf{q}_2 = \{\mathbf{w}_2, \mathbf{v}_2\}$ as

$$d = \frac{1}{2}(\mathbf{q}_1 - \mathbf{q}_2)^T H (\mathbf{q}_1 - \mathbf{q}_2)\,, \tag{2.44}$$

where for H we take the Hessian of minus the log-likelihood, estimated in $\{\mathbf{w}^{\mathrm{ML}}, \mathbf{v}^{\mathrm{ML}}\}$. In each step of the elimination process, for each parameter w_{D} we calculate the distance between the current model and the model without w_{D} that is closest to the original model. The reduced model with the smallest distance d is selected and the corresponding parameter w_{D} eliminated.

2.9 Properties of the reduced network

In Section 2.8 we have defined two backward elimination algorithms, one for the Bayesian approach, one for the ML Cox method. In this section we will apply these algorithms to the model parameters found in Section 2.7, after which we will make the final comparison between the Bayesian approach and the ML Cox method.

Figure 2.6 demonstrates the effect of the backward elimination process on the predictive power of the model. It can be seen that this procedure indeed has a wholesome effect: in both the ML Cox method and the Bayesian approach the test error decreases when the least relevant inputs are removed. The decrease of test error in the Cox method is larger than in the variational approach, since in the latter most of the overfitting problem has already been eliminated by the Bayesian priors; in the Cox method it still has to be removed through elimination of irrelevant variables. However, even after reduction the Bayesian approach still yields significantly better results than ML Cox. The effect of reducing the size of the database, described for the complete model in Section 2.7.1, does not change for the reduced model: for smaller databases the error in the ML Cox model increases significantly, whereas the Bayesian method remains much more stable. The variational approach and the Laplace approximation yield similar results, although the Laplace approximation tends to yield slightly better predictions at some points, including the optimal number of parameters. This difference may be due to the fact that the variance for the Laplace approximation is unconstrained, whereas the variance in the variational approach has a very specific form (see Equations 2.24 and 2.25).

It can also be seen (Figure 2.6, lower panel) that the Bayes factor gives a good estimation of the point where further reduction is no longer desirable: on average, the lowest test error is realized when 4 inputs are left in the network. The Bayes factor drops below one and stops the process when 2-4 inputs are left, which is clearly within the minimal test error range.

Although backward elimination is considered a rather instable procedure (see e.g. [14]), in the Bayesian approach the individual results do not vary

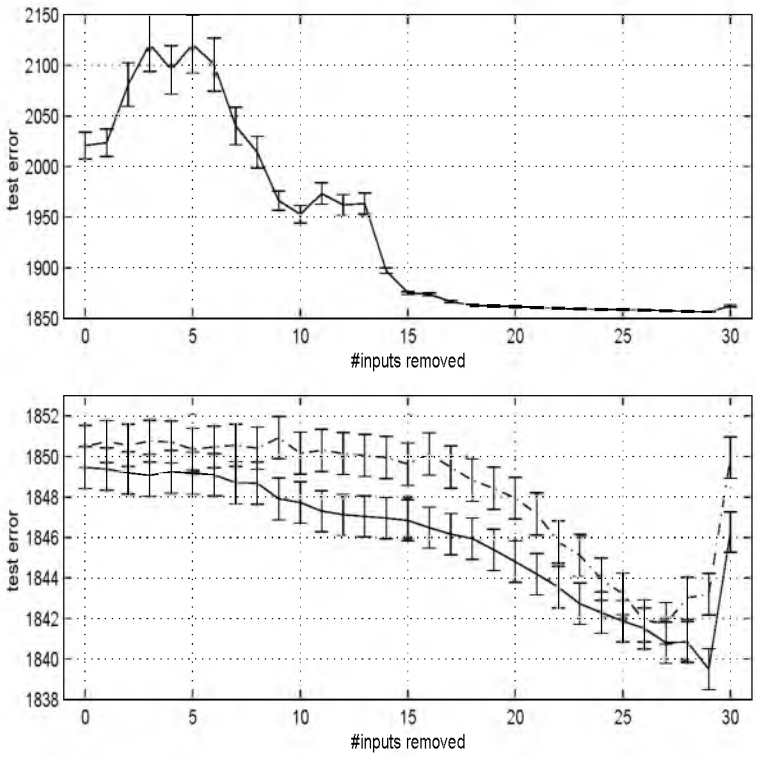


Figure 2.6: The test error (minus the log-likelihood of the data in the test set under the current model) as a function of the number of removed input parameters for the maximum likelihood Cox model (upper), the variational approach (lower, dashed line) and the Laplace approximation (lower, solid line). At zero, thirty one inputs are left in the model, at thirty, just one. The test error is the average error over 25 sessions, conducted on parameters obtained from random choices of training sets, each containing 600 patterns.

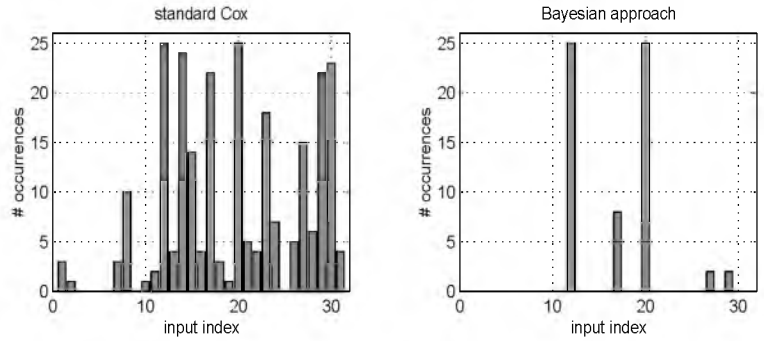


Figure 2.7: For the Bayesian approach (right panel), in each of the 25 model reduction sessions 2-4 inputs were left in the model. Two of the inputs (12 and 20) appear in almost all of the results. Inputs 17, 27 and 29 are less strong, but still clearly present in the results. For the ML Cox method (left panel), the reduction process is considerably less stable, resulting in a more noisy selection of remaining parameters.

dramatically: backward elimination on different training sets generally yielded the same set of remaining parameters. Reducing the size of the database did not have any significant effect on the choice of remaining parameters. The number of times each input remained after the elimination process is indicated in Figure 2.7. In the Bayesian approach three inputs (patient's performance, leucocytes and the number of tumors after surgery) are found to be extremely relevant. In the ML Cox method the selection of remaining parameters varies strongly between different training sets.

2.10 Alternative methods

Many other neural approaches to survival analysis have been proposed. One approach is to implement a series of classification problems, one for each discrete point in time [55, 70, 81]. For each of the problems the task of the network is to discriminate between patients who survive up to the corresponding time, and patients who do not. In this setting it may be hard to ensure the consistency of the model (i.e. if a patient is predicted to survive up to some time t a model corresponding to an earlier time should predict the same). Another problem in this approach is the treatment of censored patients.

An alternative to splitting the survival analysis problem into multiple classification problems is to use time as an input [9, 27]. An argument in favor of this method is that time can be treated as a continuous parameter, yielding higher accuracy. We found however (Section 2.7.3) that a finer discretization

of our hazard has no significant positive effect on the models accuracy. A disadvantage of using time as an input is that time, although it plays a completely different role in survival analysis, is treated on the same footing as explanatory variables such as patient characteristics.

An argument against the use of proportional hazards is that a model with a higher degree of complexity may describe the survival process better. We showed however (Section 2.7.2) that adding more hidden units had no significant effect on the Bayesian approach (HMCMC) and made the ML Cox method perform worse, even on a large database. These results are in agreement with the earlier findings of Ripley *et al* [70].

In the existing literature, the weak points of Cox analysis we have considered in this chapter, overfitting of the proportional hazard and irregularity of the baseline hazard, have been acknowledged. Solutions have been proposed by Biganzoli [9], who added weight decay on the input covariates $\mathbf{w}$, while the problem of irregularity of the outputs has been approached by Liestöl [55], who imposed a penalty term on the difference between the baseline hazard parameters. Another often used approach to solve the latter problem is to use cubic splines methods [38, 43, 46].

2.11 Discussion

We have demonstrated that survival analysis can benefit strongly from a Bayesian approach, in particular for small data sets which are typically encountered in practice. The resulting method remains as clear and easy to use as the original Cox model, yet at the same time it is more robust and reliable.

We chose to hold on to the Cox proportional hazard model, which is still the standard in the field of survival analysis. The problems that are connected to the proportional hazards method are elegantly solved by embedding proportional hazards in a Bayesian framework. By choosing the right priors, we eliminated the problem of overfitting on the training data, resulting for example in a less irregular baseline hazard and obtained better survival predictions. This improvement was particularly clear for relatively small databases.

Within the Bayesian treatment we reviewed three methods to approximate the posterior: HMCMC sampling, the variational approach and the Laplace approximation. The results showed that sampling outperformed the other two methods in predictive qualities, while there was no significant difference between variational methods and Laplace. However, HMCMC sampling not only takes more time, but is also cumbersome to use in practice: backward elimination for example, can be done directly when the posterior is described by an approximating normal distribution, yet is not easy to do in the sampling approach.

Since it is well known that the Cox proportional hazards method produces

poor results with large numbers of inputs, we designed a backward elimination procedure based on the obtained approximations to the Bayesian posterior. Although removal of irrelevant inputs indeed greatly improved the ML Cox method, its predictive qualities still did not exceed those of the Bayesian approach with a full set of inputs. Comparison between the reduced ML Cox model and the reduced Bayesian model again proved the latter to yield significantly better results. Furthermore, the selection of 'relevant' inputs was shown to be much more stable under the Bayesian approach than for the ML Cox method.

The Bayesian framework which has been presented in this chapter provides a solid basis for survival analysis, yet there is still room for further research. For example, although our extension of the model to non-proportional hazards did not yield a significant improvement, another more complicated model might yield different results. Another opening is the fact that on our database the variational approach and the Laplace approximation achieved similar results, even though the variational approach is more sophisticated. It would be interesting to see whether, on a more complicated database, this equality still holds.

Appendix 2.A Calculation of the Bayes factor

In Section 2.8 we expressed the Bayes factor as

$$BF = \lim_{\epsilon \to 0} \frac{P(|w_{\mathrm{D}}| < \epsilon | D)}{P(|w_{\mathrm{D}}| < \epsilon)} \,. \tag{2.45}$$

The lower term can be obtained from the prior on $\mathbf{w}$:

$$\begin{aligned} P(|w_{\mathrm{D}}| < \epsilon) &= \int d\lambda \; P(|w_{\mathrm{D}}| < \epsilon | \lambda) P(\lambda) \\ &= \int d\lambda \int d\mathbf{w}_{\mathrm{R}} \int_{-\epsilon}^{\epsilon} dw_{\mathrm{D}} \left[\frac{\lambda^n |\Lambda|}{(2\pi)^n} \right]^{1/2} \exp\left[-\frac{\lambda}{2} \mathbf{w}^T \Lambda \mathbf{w} \right] P(\lambda | \sigma, \tau) \\ &= 2\epsilon \left[\frac{|\Lambda|}{2\pi |\Lambda_{RR}|} \right]^{1/2} \frac{\tau^{\sigma}}{\Gamma(\sigma)} \frac{\Gamma(\hat{\sigma})}{\tau^{\hat{\sigma}}} + \mathcal{O}(\epsilon^2) \end{aligned} \tag{2.46}$$

where n is the dimension of $\mathbf{w}$, $\hat{\sigma} = \sigma + \frac{1}{2}$ and $\mathcal{O}(\epsilon^2)$ indicates terms of order ϵ^2 and smaller.

To calculate the upper term, we use the variational approximation described in Section 2.6.3:

$$\begin{aligned} P(|w_{\mathrm{D}}| < \epsilon | D) &\approx \int d\mathbf{v} \int d\mathbf{w}_{\mathrm{R}} \int_{-\epsilon}^{\epsilon} dw_{\mathrm{D}} Q(\mathbf{w}, \mathbf{v}) \qquad (2.47) \\ &= 2\epsilon \left[2\pi |C_{ww,\mathrm{DD}}| \right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \hat{w_{\mathrm{D}}}^T C^{-1}_{ww,\mathrm{DD}} \hat{w}_{\mathrm{D}} \right) + \mathcal{O}(\epsilon^2) \,, \end{aligned}$$

where $C_{ww,\mathrm{DD}}$ is the variance of w_D. Note that both the numerator and the denominator of the Bayes factor contain a factor ϵ, which therefore drops out of the equation.

Appendix 2.B Recalculation of the posterior

The posterior probability of the remaining parameters $\mathbf{v}$ and $\mathbf{w}_\mathrm{R}$ after deletion of the parameter w_D reads:

$$\begin{aligned} P(\mathbf{w}_\mathrm{R}, \mathbf{v}|w_\mathrm{D}=0, D) &= \frac{P(\mathbf{w}_R, w_\mathrm{D}=0, \mathbf{v}|D)}{P(w_\mathrm{D}=0|D)} \\ &\propto P(\mathbf{w}_R, w_\mathrm{D}=0, \mathbf{v}|D) \\ &\approx Q(\mathbf{w}_\mathrm{R}, w_\mathrm{D}=0, \mathbf{v}) \\ &\propto \mathcal{N}(\hat{\mathbf{w}}', \hat{\mathbf{v}}, C'), \end{aligned} \tag{2.48}$$

where in the second line we dropped the constant $P(w_\mathrm{D}=0|D)$ and in the third line we inserted the variational approximation. The parameters of the resulting normal distribution are defined through

$$\left[C'_{ww}\right]^{-1} = [C_{ww}]^{-1}_{\mathrm{RR}} \tag{2.49}$$

$$C'_{vv} = C_{vv} \tag{2.50}$$

$$\hat{\mathbf{w}}' = \hat{\mathbf{w}}_\mathrm{R} + C'_{ww}\left[C^{-1}_{ww}\right]_{\mathrm{RD}} \hat{w}_\mathrm{D}\,, \tag{2.51}$$

where $\left[C^{-1}_{ww}\right]_{\mathrm{RD}}$ is the block in the matrix C^{-1}_{ww} indicated by the indices of $\mathbf{w}_\mathrm{R}$ and w_D in $\mathbf{w}$ (and similar for $\left[C^{-1}_{ww}\right]_{\mathrm{RR}}$).

Chapter 3

Task clustering and gating for Bayesian multitask learning

Abstract Modeling a collection of similar regression or classification tasks can be improved by making the tasks 'learn from each other'. In machine learning, this subject is approached through 'multitask learning', where parallel tasks are modeled as multiple outputs of the same network. In multilevel analysis this is generally implemented through the mixed-effects linear model where a distinction is made between 'fixed effects', which are the same for all tasks, and 'random effects', which may vary between tasks. In this chapter we will adopt a Bayesian approach in which some of the model parameters are shared (the same for all tasks) and others more loosely connected through a joint prior distribution that can be learned from the data. We seek in this way to combine the best parts of both the statistical multilevel approach and the neural network machinery. The standard assumption expressed in both approaches is that each task can learn equally well from any other task. In this chapter we extend the model by allowing more differentiation in the similarities between tasks. One such extension is to make the prior mean depend on higher-level task characteristics. More unsupervised clustering of tasks is obtained if we go from a single Gaussian prior to a mixture of Gaussians. This can be further generalized to a 'mixture-of-experts' architecture with the gates depending on task characteristics. All three extensions are demonstrated through application both on an artificial data set and on two real-world problems, one a school problem and the other involving single-copy newspaper sales.

Adapted from: B. Bakker and T. Heskes. Task Clustering and Gating for Bayesian Multitask Learning, *Journal of Machine Learning Research*, 4:83-99.

3.1 Introduction

Many real-world problems can be seen as a series of similar, yet self contained tasks. Examples are the school problems (see e.g. [2]), and clinical trials. The first example deals with the prediction of student test results for a collection of schools, based on school demographics. The similar tasks in the other example can be the prediction of survival of patients in different clinics (see e.g. [26]). The relatedness (and therefore interdependency) of such models is taken into account and benefitted from in the fields of multitask learning (or learning to learn) and 'multilevel analysis'. In this chapter we seek to combine insights that are obtained in the field of multilevel analysis with methods that have been designed in the neural network community to create a synergetic new approach.

Multilevel analysis is generally based on the 'mixed-effects linear model'. This model features a response that is made up from the sum of a fixed effect and a random effect. The fixed effect implements a 'hard sharing' of parameters, whereas the random effect implies a 'soft sharing' through the use of a common distribution for certain model parameters. A more elaborate description of multilevel analysis is given in Section 3.6. A neural network model would use 'hard shared' parameters (the same for each of the parallel tasks) to detect 'features' in the covariates $\mathbf{x}$, and use these features for regression [7, 23]. Feature detection can be implemented, for example, in the hidden layers of a multi-layered perceptron, or through principal component analysis. This use of features is appropriate when the covariates are relatively high-dimensional, as they are for the real-world problems addressed in this chapter.

In our approach, all shared parameters, including but not restricted to the hyperparameters specifying the prior, are inferred through a maximum likelihood procedure: they are 'learned' from the data. This type of optimization has previously been studied by Baxter [7]: he showed that the risk of overfitting the shared parameters is an order N (the number of tasks) smaller than overfitting the task-specific parameters (hidden-to-output weights). The remaining parameters specific to each task are treated in a Bayesian manner. In the multitask setting, where data from all tasks can be used to fit the shared parameters, this empirical Bayesian approach is a natural choice and a close approximation to a full Bayesian treatment.

Any multitask learning model makes use of the fact that the tasks (the parallel sets of responses and covariates) are somehow related. Although for particular sets of tasks the nature of this relationship may be immediately clear, on other occasions more subtle relations may exist. Even if all tasks in a set are related, some may be stronger related to each other than to others. We accommodate for this possibility by suggesting a form of 'task clustering' (Section 3.4). Allowing a fixed number of task clusters we are able to obtain

better estimates for the responses y, and discern hidden structure within the set of tasks.

We will describe the general structure of the multitask learning model in Section 3.2. We present our Bayesian treatment of multitask learning and knowledge sharing in Section 3.3, and show how to optimize the shared parameters of the model. In Section 3.4 we extend the method so that it may allow a distinction between (groups of) tasks. The model is tested (Section 3.5) both on an artificial data set, which consists of samples drawn from a mixture of Gaussians, and on two real-world data sets: the Junior School Problem (predicting test results for British school children) and the Telegraaf problem (predicting newspaper sales in The Netherlands). We show that the method presented in this chapter yields both better predictions and a meaningful clustering of the data. Section 3.6 describes the links of the contents of this chapter with related work. We finish with concluding remarks and an outlook on future work in Section 3.7.

3.2 A neural network model

Suppose that for task i we are given a data set $D_i = \{\mathbf{x}_i^\mu, y_i^\mu\}$, with $\mu = 1 \ldots n_i$, the number of examples for task i. For notational convenience, we assume that the response y_i^μ is one-dimensional. The input $\mathbf{x}_i^\mu$ is an n_{input}-dimensional vector with components x_{ik}^μ. Our model assumption is that the response y_i^μ is the output of a multi-layered perceptron with additional Gaussian noise of standard deviation σ (see also Figure 3.1). Each output unit represents the response for one task.

Throughout this chapter we will use networks with one layer of hidden units, with either linear or nonlinear (tanh) transfer functions, and bias. The transfer functions on the outputs will be linear. The bottom layer of this network creates that lower-dimensional representation (see e.g. [7]) of the inputs, that is best suited for the second layer to perform regression on.

In our model, the input-to-hidden weights W are shared by (equal for) all tasks, whereas the hidden-to-output weights are task-dependent (see Figure 3.1). In this format the expression for the response y_i^μ reads:

$$y_i^\mu = \sum_{j=1}^{n_{\text{hidden}}} A_{ij} h_{ij}^\mu + A_{i0} + \text{noise} \,, \quad h_{ij}^\mu = g\left(\sum_{k=1}^{n_{\text{input}}} W_{jk} x_{ik}^\mu + W_{j0} \right) \,, \quad (3.1)$$

with W the $n_{\text{hidden}} \times (n_{\text{input}} + 1)$ matrix of input-to-hidden weights (including bias) and $\mathbf{A}_i$ an $(n_{\text{hidden}} + 1)$-dimensional vector of hidden-to-output weights (and bias). The extra index i in the hidden unit activity $\mathbf{h}_i^\mu$ follows from the dependency of the covariates $\mathbf{x}_i^\mu$ on task i. For notational simplicity we will include $h_{i0}^\mu = 1$ in $\mathbf{h}_i^\mu$ and A_{i0} in $\mathbf{A}_i$ from now on.

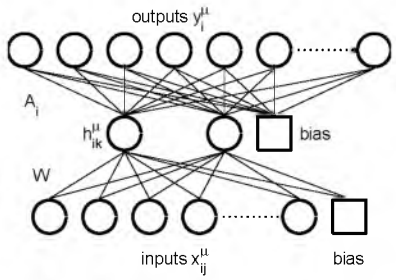


Figure 3.1: Neural network model. The input-to-hidden weights W are shared by (equal for) all tasks. Each of the outputs (top layer) represents one task, and has its own set of task-dependent hidden-to-output weights A_i.

3.3 Computation and optimization

Let us now consider the full set of tasks, for which we define the complete data set $D = \{D_i\}$, with $i = 1 \dots N$, the number of tasks. For notational convenience we will assume all inputs $\mathbf{x}_i^\mu$ fixed and given and omit them from our notation. We assume the tasks to be iid given the hyperparameters, and define a prior distribution for the task-dependent parameters:

$$\mathbf{A}_i \sim \mathcal{N}(\mathbf{A}_i|\mathbf{m}, \Sigma) \,, \tag{3.2}$$

which is a Gaussian with an $(n_{\text{hidden}}+1)$-dimensional mean $\mathbf{m}$ and an $(n_{\text{hidden}}+1) \times (n_{\text{hidden}}+1)$ covariance matrix Σ.

This prior distribution is incorporated into the posterior probability of data and hidden-to-output weights given the hyperparameters $\Lambda = \{W, \mathbf{m}, \Sigma, \sigma\}$. The joint distribution of data and model parameters reads

$$P(D_i, A_i|\Lambda) = P(D_i|\mathbf{A}_i, W, \sigma) P(\mathbf{A}_i|\mathbf{m}, \Sigma) \;\forall_i \,. \tag{3.3}$$

Integrating over A we obtain, after some calculations (see Appendix 3.A for details)

$$P(D|\Lambda) = \prod_{i=1}^{N} P(D_i|\Lambda) \,, \tag{3.4}$$

with

$$P(D_i|\Lambda) \propto \left(|\Sigma|\sigma^{2n_i}|Q_i|\right)^{-\frac{1}{2}} \exp\left[\frac{1}{2}(\mathbf{R}_i^T Q_i^{-1}\mathbf{R}_i - S_i)\right] , \tag{3.5}$$

where Q_i, $\mathbf{R}_i$ and S_i are functions of D_i and Λ given by

$$\begin{aligned} Q_i &= \sigma^{-2} \sum_{\mu=1}^{n_i} \mathbf{h}_i^\mu \mathbf{h}_i^{\mu T} + \Sigma^{-1} \;, \quad \mathbf{R}_i = \sigma^{-2} \sum_{\mu=1}^{n_i} y_i^\mu \mathbf{h}_i^\mu + \Sigma^{-1} \mathbf{m} \;, \\ S_i &= \sigma^{-2} \sum_{\mu=1}^{n_i} (y_i^\mu)^2 + \mathbf{m}^T \Sigma^{-1} \mathbf{m} \;. \end{aligned} \tag{3.6}$$

(Recall that $\mathbf{h}_i^\mu$ depends on W and $\mathbf{x}_i^\mu$.) In (3.4) we used the assumption that the tasks are iid given Λ. Optimal parameters Λ^* are now computed by maximizing the likelihood (3.5). This is referred to as empirical Bayes [71], and is similar to MacKay's evidence framework [59]. Here it is motivated by the fact that we can use the data of all tasks to optimize Λ, the dimension of which is independent of and assumed to be much smaller than the number of tasks. In the limit of infinite tasks the empirical Bayesian approach coincides with the full Bayesian approach. For finite numbers of tasks, Baxter [7] shows that the generalization error as a function of the number of tasks N and the dimension of the hyperparameters $|\Lambda|$ is proportional to $|\Lambda|$ and inversely proportional to N (see also [41]).

Given the maximum likelihood parameters Λ^*, we can easily compute $P(\mathbf{A}_i|D_i, \Lambda^*)$ to make predictions, compute error bars, and so on. The parameter

$$\tilde{\mathbf{A}}_i = \operatorname*{argmax}_{\mathbf{A}_i} P(\mathbf{A}_i|D_i, \Lambda^*) \tag{3.7}$$

will be referred to as the maximum *a posteriori* or MAP value for $\mathbf{A}_i$.

When $g(\cdot)$ in (3.1) is a linear function, we can simplify Equation 3.5 significantly (see Appendix 3.A). This gives us the advantage that instead of using the full data sets $D_i = \{\mathbf{x}_i^\mu, y_i^\mu\}$ for each task, we only need the sufficient statistics $\langle \mathbf{x}_i \mathbf{x}_i^T \rangle$, $\langle \mathbf{x}_i y_i \rangle$ and $\langle y_i^2 \rangle$ for optimization, where $\langle .. \rangle$ denotes the average over all examples μ.

3.4 Making some tasks more similar than others

The prior (3.2) may be useful when we have reason to believe that *a priori* all tasks are 'equally similar'. In many applications, this assumption is a little too simplistic and we have to consider more sophisticated priors.

3.4.1 Task-dependent prior mean

We can make the prior distribution task-dependent by introducing higher-level task characteristics, that is, 'features' of the task that are known beforehand. We will denote them $\mathbf{f}_i$ for task i. These features, although they have different

values for different tasks, do not vary within one task. Therefore, rather than adding them as extra inputs, we use these features to make the prior mean task-dependent. A straightforward way to include these features into the prior mean is to make it a linear function of these features, that is,

$$\mathbf{m}_i = M\mathbf{f}_i \,, \tag{3.8}$$

with M now an $(n_{\text{hidden}}+1) \times n_{\text{feature}}$ matrix. We are back to the independent prior mean if we take $n_{\text{feature}} = 1$ and all $f_i = 1$.

The calculation of the likelihood proceeds as in Section 3.3, with $\mathbf{m}$ in (3.6) replaced by $\mathbf{m}_i$. With a linear form optimization is hardly more involved, but in principle we can take more complicated nonlinear dependencies into account as well.

3.4.2 Clustering of tasks

Another reasonable assumption might be that we have several clusters of similar tasks instead of a single cluster. Then we could take as a prior a mixture of n_{cluster} Gaussians,

$$\mathbf{A}_i \sim \sum_{\alpha=1}^{n_{\text{cluster}}} q_\alpha \mathcal{N}(\mathbf{m}_\alpha, \Sigma_\alpha) \tag{3.9}$$

instead of the single Gaussian (3.2). Each Gaussian in this mixture can be seen to describe one 'cluster' of tasks. In Equation 3.9, q_α represents the *a priori* probability for any task to be 'assigned to' cluster α (see also Figure 3.2). Although *a priori* we still do not distinguish between different tasks, *a posteriori* tasks can be assigned to different clusters. The posterior data likelihood reads:

$$P(D_i|\Lambda) = \int d\mathbf{A}_i P(D_i|\mathbf{A}_i, \Lambda) \sum_{\alpha=1}^{n_{\text{cluster}}} q_\alpha P(\mathbf{A}_i|\mathbf{m}_\alpha, \Sigma_\alpha) \,. \tag{3.10}$$

The major 'probability mass' of this integral lies in areas where the parameters $\mathbf{A}_i$ both lead to high probabilities of the data (are able to fit the data well) and have high probability under $P(\mathbf{A}_i|\mathbf{m}_\alpha, \Sigma_\alpha)$ themselves. In this way, the posterior distribution effectively 'assigns' tasks to that cluster that is most compatible with the data within the task, in the sense that all other clusters (Gaussians) do contribute much less to (3.10).

We introduce indicator variables $z_{i\alpha}$ where $z_{i\alpha}$ equals one if task i is assigned to cluster α and zero otherwise. For any task i only one $z_{i\alpha}$ may be one. To optimize the likelihood of the shared parameters Λ, which now include all cluster means and covariances as well as the prior assignment probabilities $\mathbf{q}$, we can apply an expectation-maximization or EM algorithm (see e.g. [28]).

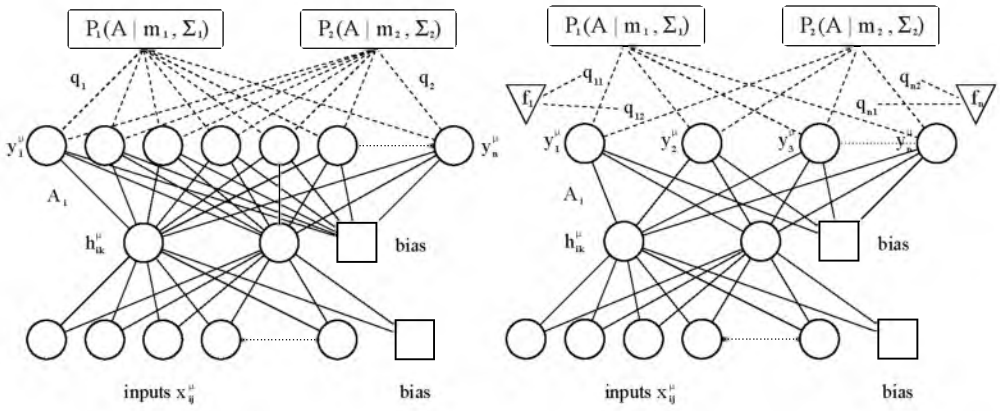


Figure 3.2: The task clustering (left) and task gating model (right). In task clustering the task-dependent weights $\mathbf{A}_i$ are supposed to be drawn from a weighted sum of Gaussians, where the weights q_α are equal for all tasks. In task gating these weights become task-dependent and the value for $q_{i\alpha}$ depends on the task-specific feature vector $\mathbf{f}_i$.

If the cluster assignments $z_{i\alpha}$ were known, optimization of $\log P(D, z|\Lambda)$ with respect to Λ would be relatively simple. The values for $z_{i\alpha}$ however are not known, so in the E-step we *estimate* the expectation value of $\log P(D, z|\Lambda)$ under $P(z|D, \Lambda_n)$, where for Λ_n we take the current values for Λ (which are initialized randomly at the start of the procedure). In the M-step the obtained expectation value is maximized for Λ. This step is of roughly the same complexity as the optimization for a single cluster. Both steps are repeated until Λ converges to a (local) optimum. This implementation of the EM algorithm is described in more detail in Appendix 3.B.

3.4.3 Gating of tasks

A possible disadvantage of the above task clustering approach is that the prior is task-independent: *a priori* all tasks are assigned to each of the clusters with the same probabilities q_α. A natural extension is to incorporate the task-dependent features $\mathbf{f}_i$ that were introduced in 3.4.1 in a gating model (see Figure 3.2), for example by defining

$$q_{i\alpha} = \mathrm{e}^{\mathbf{U}_\alpha^T \mathbf{f}_i} / \sum_{\alpha'} \mathrm{e}^{\mathbf{U}_{\alpha'}^T \mathbf{f}_i} , \tag{3.11}$$

with $\mathbf{U}_\alpha$ an n_{feature}-dimensional vector. The *a priori* assignment probability of task i to cluster α is now task-dependent. $\mathbf{U}_\alpha$ performs a similar function as the matrix M in Section 3.4.1: for each task, it translates the task-dependent

feature vector $\mathbf{f}_i$ to a preference for one or more of the clusters α. $\mathbf{U}_\alpha$ is added to the set of hyperparameters Λ and learned from the data.

The above task clustering approach is a special case with $n_{\text{feature}} = 1$ and $\mathbf{f}_i = 1$ for all tasks i. The EM algorithm is similar to the one described in Appendix 3.B: we simply replace q_α with $q_{i\alpha}$. The M-step for the parameters $\mathbf{U}_\alpha$ becomes slightly more complicated, and can be solved using an iterative reweighted least-squares (IRLS) algorithm. The task gating part of our model can be compared to the mixture of experts model [52]. An important difference is that we use a separate set of higher-level features to gate tasks rather than individual inputs.

3.5 Results

We tested our method on three databases, which are described in the following paragraphs. For the first database we implemented neural networks that feature one hidden unit with either a hyperbolic tangent or a linear transformation as a transfer function. For the other databases we implemented networks with two hidden units, and hyperbolic tangents as transfer functions. Each network features linear output units. Networks with more (or less) hidden units did not significantly improve prediction. Each hidden and each output unit contains an additive bias (see also Figure 3.1). For each dataset we also consider the performance of the single task learning method (training a separate neural network for each task). For the school and the newspaper data, we also look at non-Bayesian multitask learning: in this intermediate model we applied the same network structure as in the Bayesian multitask learning model, yet instead of estimating a prior distribution we learned all model parameters directly. For these two non-Bayesian methods we applied early stopping (see e.g. [24]) to prevent overfitting on the training data (the model parameters were optimized on a training set, and the optimization process stopped when no more improvement was found on a separate validation set).

3.5.1 Description of the data

Artificial Data. We created a data set of artificial data, by drawing random covariates $\mathbf{x}_i^\mu$ and shared parameters Λ (see Appendix 3.C for the exact values.). The $\mathbf{x}_i^\mu$ were scaled per task to have zero mean and unit variance. The responses y_i^μ were drawn according to Equation 3.1, where we used a generative model with one hidden unit and two clusters (two choices for $\mathbf{m}_\alpha$). The artificial data did not include task-dependent features $\mathbf{f}_i$. We studied both data sets for $g(x) = \tanh(x)$, and $g(x) = x$. To test our methods we ran 10 independent simulations, where each time we used a random selection of 10 covariates and their corresponding responses per task for optimization, and a

large independent test set (300 samples per task) to check the performance of the model. In each simulation, we used 250 parallel tasks.

School Data. This data set, made available by the Inner London Education Authority (ILEA), consists of examination records from 139 secondary schools in years 1985, 1986 and 1987. It is a random 50% sample with 15362 students. The data set has been used to study the effectiveness of schools. A file containing the database can be downloaded from the 'Multilevel Page' (`http://multilevel.ioe.ac.uk/intro/datasets.html`). See also [64].

Each task in this setting is to predict exam scores for students in one school, based on eight inputs. The first four inputs (year of the exam, gender, VR band and ethnic group) are student-dependent, the next four (percentage of students eligible for free school meals, percentage of students in VR band one, school gender (mixed or (fe)male only) and school denomination) are school-dependent. The categorical variables (year, ethnic group and school denomination) were split up in binary variables, one for each category, making a new total of 16 student-dependent inputs, and six school-dependent inputs. We scaled each covariate and output to have zero mean and unit variance. All performance measures are obtained after making 10 independent random splits of each school's data (covariates and corresponding responses) into a 'training set' (containing on average 80 samples), used both for fitting the shared parameters and computing the MAP hidden-to-output weights, and a 'test set' (comprised of the remaining samples, 30 on average) for assessing the generalization performance.

Prediction of Newspaper Sales. We also applied our methods on a database of single-copy sales figures for one of the major Dutch newspapers. The database contains the numbers of newspapers sold on 156 consecutive Saturdays, at 343 outlets in The Netherlands. Inputs include recent sales (four to six weeks in the past), last year's sales (51 to 53 weeks in the past), weather information (temperature, wind, sunshine, precipitation quantity and duration) and season (cosine and sine of scaled week number). The responses are the realized sales figures. Considering a single task, our model can be interpreted as an auto-regressive model with additional covariates. All covariates and responses were scaled per task (outlet) to zero mean and unit variance. Performance measures were obtained as for the school data, where now the 'training set' contains 100 samples per task and the 'test set' contains the remaining 56 samples. For the task-dependent mean and the gating of tasks we constructed two features depending on the outlet's location: the first feature codes the number of local inhabitants (from zero, less than 15,000, to four, more than 300,000), the second one the level of tourism (from zero, hardly any tourism, to two, very touristic).

Table 3.1: Explained variance for each of the combinations of model and database.

	single cluster	task clustering
linear model on linear data	40.5 ± 0.5 %	43.3 ± 0.5 %
linear model on nonlinear data	22.0 ± 0.3 %	23.8 ± 0.9 %
nonlinear model on linear data	40.9 ± 0.4 %	40.9 ± 0.5 %
nonlinear model on nonlinear data	23.3 ± 0.5 %	24.9 ± 0.6 %

3.5.2 Generalization performance

Artificial Data. We applied both the linear and the nonlinear method to the two databases we created, resulting in four combinations. In each of these four combinations we applied both the multitask learning method with one cluster, and model clustering. As can be seen in Figure 3.3, in all four cases the method was able to discern the two clusters. Note that in the second panel (linear model working on nonlinear data) one of the priors is 'flattened', indicating that in this case only part of the (nonlinear) structure could be found.

Table 3.1 presents a model evaluation in terms of the percentage of variance explained by both models with and without clustering. Percentage explained variance is defined as the total variance of the (combined training and test) data minus the average squared error on the test set, expressed as a percentage of the total data variance. All combinations of model and database except the nonlinear model on the linear database showed significant improvements when task clustering is implemented. Note also that nonlinear multitask learning on the linear database performed equally well as the linear method. Apart from this, the nonlinear model worked best for the nonlinear database and the linear model for the linear database. For both databases, single task learning explained less than one percent of the variance.

School Data. We applied single task learning, non-Bayesian and Bayesian multitask learning on the school data. The results are expressed in Table 3.2. Single task learning explained 9.7% of the variance, which was much less than any of the multitask learning methods. Non-Bayesian multitask learning explained 29.2% of the variance. The overall winners were the Bayesian methods with one and two priors (clusters) with an explained variance of 29.5%. Implementation of the methods described in Section 3.4 yielded no improvement on the 'single cluster' multitask learning method. This lack of improvement was also reflected in the clusters obtained: either two very similar priors were created, or one cluster was found to contain all tasks, whereas the other was empty. Although task clustering did not yield a significant improvement here,

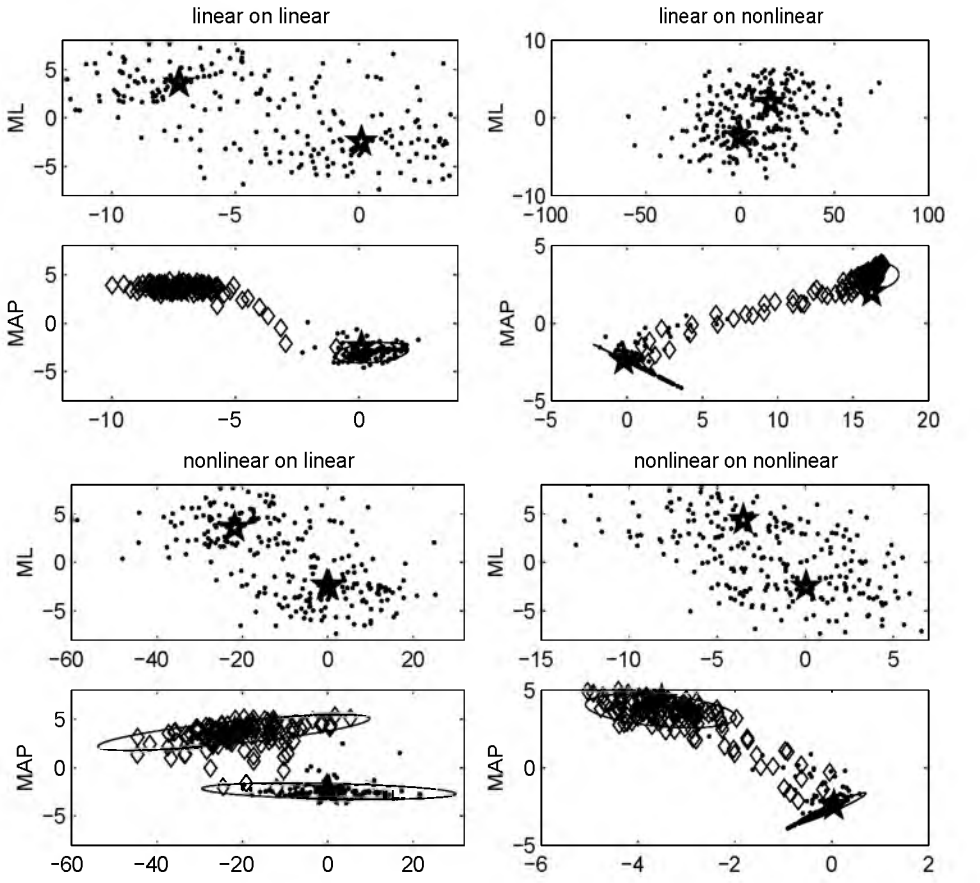


Figure 3.3: Maximum likelihood (upper panels, marked ML) and maximum *a posteriori* (lower panels, marked MAP) values for the model parameters A_i in the artificial data paradigm. Each dot or diamond refers to one task, where (in the lower panels) identical marks (dots or diamonds) indicate tasks that belong to the same cluster (are generated around the same mean). In each panel, the horizontal axis corresponds to the hidden-to-output weight, the vertical axis to the bias. The 95% confidence intervals for the two estimated priors are plotted in the lower panels. From left to right and up to down, the panels show the results for the linear model on the linear data, the same model on the nonlinear data, the nonlinear model on the linear data and on the nonlinear data. The means m_α used for generation of the data sets (corrected for the difference between the true and estimated hidden unit activity) are depicted by stars in the 8 panels.

Table 3.2: Explained variance for the school data and the newspaper data. The evaluated methods are single task learning (STL), maximum likelihood multitask learning (ML MTL), Bayesian multitask learning, task clustering with two clusters and task gating with two clusters. Task gating is not applied to the school data.

	school data	newspaper sales
STL	9.7 ± 0.7 %	< 1 %
ML MTL	29.2 ± 0.3 %	9.0 ± 0.3 %
Bayesian MTL	29.5 ± 0.4 %	11.1 ± 0.4 %
task clustering	29.5 ± 0.4 %	11.2 ± 0.4 %
task gating		11.2 ± 0.3 %

at least the results show that the method does not force structure on the data where there is no structure present.

Prediction of Newspaper Sales. The model of Section 3.3 (single Gaussian prior) managed to explain 11.1% of the variance in the test data, much better than the 9.0% explained variance with the same multitask model regularized through early stopping instead of through a 'learned' prior. Note that this regression problem has a very low signal-to-noise ratio, also due to the fact that, for a fair real-world comparison, only sales figures from at least four weeks ago can be used as covariates. When all tasks were optimized independently using all 13 covariates, less than one percent explained variance was achieved. These results are consistent with the more extended simulation studies in [40, 41].

The more involved methods of Section 3.4 all led to a slightly, but significantly better performance, explaining another 0.1% of the variance. Although not spectacular, translated to the set of more than 10,000 Dutch outlets for which predictions have to be made on a daily basis, this might still be worthwhile.

Although for each database the more involved methods (such as task clustering or gating) required more computation time than the simpler methods (such as non-Bayesian multitask learning), all times were in the same order of magnitude. None of the simulations took more than 30 minutes (on a Pentium 3). In general, when a method was able to explain a higher percentage of the variance, it also required more computation time. The one exception to this rule was the single task learning method: although it performed (relatively) poorly on all of the databases, it actually required more time than non-Bayesian multitask learning. This is caused by the fact that for single task learning the input-to-hidden weights need to be optimized for each task separately, whereas these weights are equal for all tasks in the other methods.

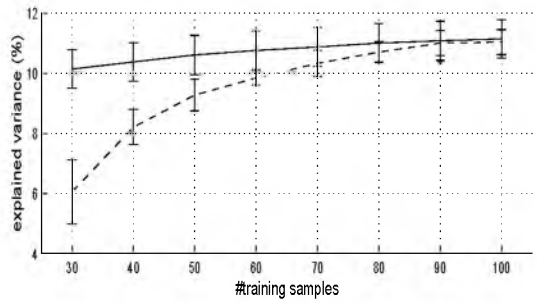


Figure 3.4: Explained variance for multitask learning on the newspaper data with one (dashed line) and two clusters (solid line): for less training samples the effect of clustering grows stronger.

More in general, multitask learning is most useful in circumstances where many parallel tasks are available, but few training samples per task. To test under such conditions, we compared the single Gaussian prior model to the model with two clusters for lower numbers of training samples per task, ranging from 30 to 100 samples. Figure 3.4 shows the explained variances for both models, which are (as before) the averages over 10 independent splits of the data into a training and a test set. The figure clearly shows that although for larger numbers of training samples the more involved model does not perform much better than the simpler model, for less training samples it yields a substantial improvement.

3.5.3 Interpretation of the newspaper results

The solutions that we obtained make sense and provide a lot of interesting information. Figure 3.5 displays a Hinton diagram [10] of the input-to-hidden weights typically and consistently (up to permutation and sign flips) found in all of the multitask learning approaches. It can be seen that one hidden unit focuses on recent sales figures (referred to as 'short term') and the other on last year's sales and season (referred to as 'seasonal').

The left panel in Figure 3.6 plots the maximum likelihood solutions for the hidden-to-output weights of the different outlets. The next panel visualizes the effect of a single Gaussian prior by plotting the corresponding MAP solutions. Task clustering yielded two distinct clusters: a 'seasonal' cluster with hardly any variation in short-term effects and a 'short-term' cluster with much less variation in the seasonal effect. The solution obtained through task gating is just slightly different. Although this particular distinction between the two clusters is not what we had expected to find (i.e. clusters with different means M_α and similar covariances Σ_α), it does make a lot of sense. Tasks in the

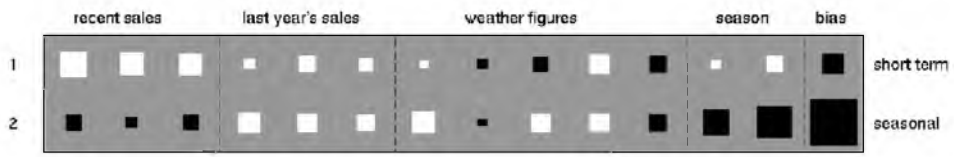


Figure 3.5: Hinton diagram of the input-to-hidden weights. Positive weights are white, negative weights are black. The absolute magnitude of each weight corresponds to the size of its square. Past sales figures are coded in the first 6 inputs, the next 5 inputs represent weather information, and the last 2 inputs indicate the season. The rightmost squares represent the biases of the hidden units.

seasonal cluster all have relatively small weights connected to the hidden unit that focuses on short term effects, yet they do display moderate to strong connections to the seasonal hidden unit. The reverse is true for the short term cluster.

The distinction between seasonal and short term tasks can be visualized on the map of the Netherlands. In Figure 3.7 on the left we marked the locations of the outlets that with weight $\rho_{i\alpha}$ larger than 0.85 are assigned to the 'seasonal' cluster and have a positive 'seasonal' hidden-to-output weight, which corresponds to higher sales in summer than in winter. Most of them are located in touristic areas (e.g. close to beaches). The right plot visualizes the outlets that with weight larger than 0.85 are assigned to the short-term cluster. These are located in the Randstad, Holland's most densely populated area, and other cities of reasonable size.

3.5.4 Difference between task clustering and task gating

It is no surprise that task gating manages to pick up the correlations between short term or seasonal effects and location of the outlet. For example, after training the *prior* assignment of an outlet in a large non-touristic city to the short-term cluster equals 0.87, whereas an outlet in a small touristic village is assigned to the seasonal cluster with weight 0.80. The reason that this prior information does not lead to (significantly) better generalization performance is that the *a posteriori* assignments based on 100 training examples appear to be of about the same quality for task clustering and gating. With less examples, the *a priori* assignments will become more important, making the gating method preferable over the clustering approach, as can be seen in Figure 3.8.

Another way to illustrate the difference between task clustering and task

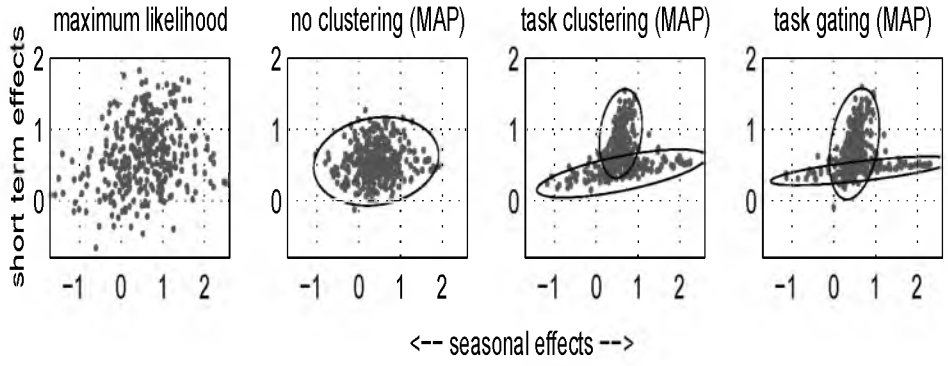


Figure 3.6: The maximum likelihood values for the hidden-to-output weights (left panel), and their MAP values (right panels). The horizontal axis of each plot refers to the hidden-to-output weight connected to the hidden unit with input-to-hidden weights that are sensitive to seasonal effects, as demonstrated in Figure 3.5. The vertical axis refers to the weights connected to the hidden unit that is sensitive to short term effects. Each mark represents the hidden-to-output weights for one task. The ellipses drawn in the right panels visualize the priors imposed on the task-dependent parameters $\mathbf{A}_i$. They indicate 95% confidence intervals of these priors. The prior depicted in the second panel is unimodal (single cluster): all task-dependent weights have the same prior distribution. In the right two panels there are two clusters, formed through task clustering (third panel) and task gating (fourth panel).

Figure 3.7: Clustering of Dutch outlets. Circles mark outlets assigned with weight larger than 0.85 to either the 'seasonal' cluster (left panel) or the 'short term' cluster (right panel).

gating is through the 'entropy' of the clusters formed by each method. We define the entropy of a set of clusters in this setting as

$$S(\rho) = -\sum_{i=1}^{N} \sum_{\alpha=1}^{n_{\text{cluster}}} \rho_{i\alpha} \log(\rho_{i\alpha}) \,, \tag{3.12}$$

where

$$\rho_{i\alpha} = P(z_{i\alpha} = 1 | \Lambda, D_i) \,, \tag{3.13}$$

the probability for task i to be in cluster α. The entropy $S(\rho)$ reaches its maximum when each task is assigned to any cluster with equal probability, and its minimum when each task is assigned to one particular cluster with probability one.

For both task clustering and gating the entropy depends on the number of samples present in the data D. For very low numbers of samples the cluster assignment $\rho_{i\alpha}$ will depend mainly on the chosen prior, whereas for larger numbers of samples the data likelihood will become the dominant factor. Figure 3.9 plots the entropy for task assignments obtained from both methods as a function of the number of samples per task. The data set used here is the Telegraaf data set, but for the purpose of this illustration it could be any data set in which a meaningful clustering can be found.

Figure 3.9 shows clearly that although for increasing numbers of samples the two entropies converge to the same value, in the case of very few samples the entropy after task gating is significantly lower. This can easily be understood by noting that for task gating the prior distribution already discriminates between tasks whereas task clustering treats all tasks equally *a priori*, resulting in a stronger predefined order for the gating method.

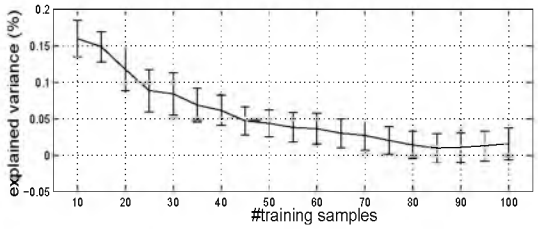


Figure 3.8: Difference in explained variance between the task gating and clustering approach: the less training samples, the better the gating method.

3.6 Related work

Multitask learning is very similar to multilevel analysis in statistics. Here, a distinction is made between level one variation between different samples within the same task (e.g. from student to student, from patient to patient) and level two variation between different tasks (e.g. between schools or hospitals). The leading model to account for such variations is the mixed-effects linear model. In this model the user is presented with a series of parallel units, each containing a set of covariates and responses. The predicted value for the μ^{th} response of unit i in this model reads

$$y_i^\mu = \mathbf{x}_i^{\mu T}\boldsymbol{\beta} + \mathbf{z}_i^{\mu T}\mathbf{b}_i\,, \tag{3.14}$$

where $\mathbf{x}_i^\mu$ is a vector of covariates, $\boldsymbol{\beta}$ is a vector of fixed effects parameters, $\mathbf{b}_i$ is the task-dependent random effect assumed to be (normal) distributed around zero with covariance Σ and $\mathbf{z}_i^\mu$ is a vector of parameters related to this effect. The parameters in this model are generally estimated through iteration of linear regression on the parameters $\boldsymbol{\beta}$, and fitting of the covariance Σ to the residuals $\tilde{y}_i^\mu = y_i^\mu - \mathbf{x}_i^{\mu T}\boldsymbol{\beta}$. An alternative to this method is the empirical Bayesian approach, where probability distributions over both $\mathbf{b}_i$ and $\boldsymbol{\beta}$ are defined. The parameters of these distributions (called hyperparameters) are optimized directly, resulting in distributions around the maximum *a posteriori* values for $\mathbf{b}_i$ and $\boldsymbol{\beta}$. Both approaches are described in [17]. In the *full* Bayesian approach (see e.g. [79]) further prior distributions are defined for these hyperparameters, which are chosen *a priori*. In this approach, however, one has to resort to sampling, which becomes infeasible for large numbers of tasks.

Over the past years many proposals have been made to incorporate non-linearity into these models, through B-splines (see e.g. [56]) and other methods [4, 16]. To the best of our knowledge, the ideas of task clustering and gating are new to this field.

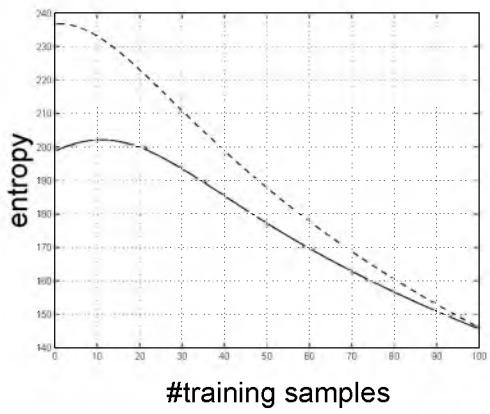


Figure 3.9: The entropy of the task-cluster assignment probabilities $\rho_{i\alpha}$ as a function of the number of samples for the task clustering method (dashed line) and the gating method (solid line). For few samples, the gating method clusters more strongly (has a lower entropy) than the task clustering method. For higher numbers of samples, the two methods behave similarly.

An alternative approach to multitask learning has been taken by Thrun and O'Sullivan [82] and Pratt [69], who have devised elegant ways to transfer knowledge obtained by one network to another network learning a similar task. Thrun and O'Sullivan [82] also suggest a task clustering algorithm, where a distance metric between parallel tasks is learned, and used to classify new learning tasks. In learning these new tasks, (only) the information contained in the corresponding cluster is exploited.

Trajectory clustering [20] can be derived as a special case of task clustering without hidden units and with all covariance matrices Σ_α set to zero. On a more philosophical level, clustering in our approach is an interesting by-product of a better generalizing model, not a goal by itself.

The gating in Section 3.4.3 yields an EM algorithm that is similar to the one for a mixture-of-experts architecture (see [51, 52]), but the application of it is quite different. In our case the gating is at the level of tasks rather than covariates and depends on task-specific properties $\mathbf{f}_i$, completely independent of the covariates $\mathbf{x}_i^\mu$.

3.7 Discussion

Our work has been inspired by Baxter [7], Caruana [23] and Thrun and O'Sullivan [82]. We have adopted the idea of recognizing parallel tasks and

learning their underlying structure from the data, and considered an extension through the use of task clustering, which has been implemented in a different form by [82]. In this chapter we implement this clustering through the design of prior distributions that are able to discriminate between tasks. The result is a hierarchical Bayesian approach to multitask learning, where some of the model parameters are shared explicitly (the input-to-hidden weights among others) and others are soft-shared through a prior distribution. These prior distributions may make use of task-dependent features which are known in advance (Sections 3.4.1 and 3.4.3), or make the distinction between tasks purely from the data samples in the training set (Section 3.4.2). The applicability of this model to real-world problems was demonstrated (among others) on the newspaper data, where we showed the usefulness of the model both in terms of explained variance and independent detection of features in the data.

Application of our methods on an artificial data set demonstrated that appropriately structured regression problems can benefit significantly both from the multitask learning approach and from task clustering. The well-known school problem was also modeled better through Bayesian multitask learning. No substructures within the collection of tasks were found however, either because they are not present at all, or because another form of (neural network) model (e.g. other transfer functions than $\tanh(x)$ or linear) is needed to exploit them. For the Telegraaf problem we found a small yet significant increase in explained variance when task clustering was applied. However, simulations for smaller numbers of training samples showed a much more substantial improvement for smaller data sets. Interesting results were obtained on a descriptive level: the model was able to make a meaningful distinction between outlets in touristic and urban areas, without being presented with this information in advance. By examining the obtained clusters more closely, a better understanding of the tasks themselves can be gained.

From a technical point of view, the empirical Bayesian approaches proposed in this chapter become feasible and tractable even for large databases because we can analytically integrate out all task-specific parameters [cf. Equation (3.5)]. For e.g. multitask classification problems, we would have to resort to appropriate approximations, perhaps similar to those used in Gaussian processes for classification [87].

Each task in the newspaper database describes a time series, where parallel tasks feature sales at parallel times. In this chapter we have made no use of any algorithm specifically designed to model such data. In the next chapter however, we further extend our model to exploit this characteristic of the data. Here we will make a connection between multilevel analysis and dynamic hierarchical models [31].

Appendix 3.A Calculation of the data likelihood

The data likelihood is calculated as follows:

$$\begin{aligned} P(D_i|\Lambda) &= \int d\mathbf{A}_i P(D_i, \mathbf{A}_i|\Lambda) \\ &= \int d\mathbf{A}_i P(D_i|\mathbf{A}_i, \Lambda) P(\mathbf{A}_i|\Lambda) \\ &\propto \sigma^{-n_i}|\Sigma|^{-1/2} \int d\mathbf{A}_i \exp\left[-\frac{1}{2}\mathbf{A}_i^T Q_i \mathbf{A}_i + \mathbf{R}_i^T \mathbf{A}_i - \frac{1}{2}S_i\right] \\ &\propto \sigma^{-n_i}|\Sigma|^{-1/2}|Q_i|^{-\frac{1}{2}} \exp\left[\frac{1}{2}(\mathbf{R}_i^T Q_i^{-1}\mathbf{R}_i - S_i)\right], \end{aligned} \tag{3.15}$$

where Q_i, $\mathbf{R}_i$ and S_i are given by

$$Q_i = \sigma^{-2}\sum_{\mu=1}^{n_i} \mathbf{h}_i^\mu \mathbf{h}_i^{\mu T} + \Sigma^{-1}\ , \quad \mathbf{R}_i = \sigma^{-2}\sum_{\mu=1}^{n_i} y_i^\mu \mathbf{h}_i^\mu + \Sigma^{-1}\mathbf{m}\ , \tag{3.16}$$

and

$$S_i = \sigma^{-2}\sum_{\mu=1}^{n_i}(y_i^\mu)^2 + \mathbf{m}^T\Sigma^{-1}\mathbf{m}\,. \tag{3.17}$$

For the linear case, the expressions for Q_i, $\mathbf{R}_i$ and S_i simplify considerably by enforcing the constraint

$$\left\langle \mathbf{h}_i \mathbf{h}_i^T \right\rangle = \left\langle W x_i x_i^T W \right\rangle = I \tag{3.18}$$

(where $\langle..\rangle$ denotes the average over all examples μ and I is the unit matrix):

$$Q_i = n_i\sigma^{-2}I + \Sigma^{-1}\ , \quad \mathbf{R}_i = \sigma^{-2}n_i W \left\langle \mathbf{x}_i y_i \right\rangle + \Sigma^{-1}\mathbf{m}\ , \tag{3.19}$$

and

$$S_i = \sigma^{-2}n_i \left\langle y_i^2 \right\rangle + \mathbf{m}^T\Sigma^{-1}\mathbf{m}\,. \tag{3.20}$$

In this case, sufficient statistics can be calculated beforehand, after which optimizing the shared parameters no longer scales with the number of tasks [41].

Appendix 3.B An EM algorithm

To optimize $\log P(D|\Lambda)$, first we add the parameter z, which refers to a particular choice of cluster assignments $z_{i\alpha}$, assigning task i to cluster α. Averaging over the distribution $P(z|\Lambda_n, D)$ given the current value of Λ, we obtain

$$\begin{aligned}
\log P(D|\Lambda) &= \log \sum_{\{z\}} P(D, z|\Lambda) \qquad (3.21)\\
&= \log \sum_{\{z\}} P(z|\Lambda_n, D) \frac{P(D, z|\Lambda)}{P(z|\Lambda_n, D)}\\
&\geq \sum_{\{z\}} P(z|\Lambda_n, D) \log P(D, z|\Lambda) - \sum_{\{z\}} P(z|\Lambda_n, D) \log P(z|\Lambda_n, D)\\
&= \sum_{\{z\}} P(z|\Lambda_n, D) \log P(D|z, \Lambda) + \sum_{\{z\}} P(z|\Lambda_n, D) \log P(z|\Lambda)\,,
\end{aligned}$$

where we dropped the negative term in the third line since it does not depend on Λ.

In the E-step we have to compute the assignments probabilities through

$$P(z_{i\alpha} = 1|\Lambda, D_i) \propto q_\alpha P(D_i|\Lambda) \qquad (3.22)$$

and sum $\log P(D|z, \Lambda)$ and $\log P(z|\Lambda)$ over all possible assignments, weighted by their probabilities. Note that if the assignments were not given, the log-likelihood of D would read

$$\log P(D|\Lambda) = \sum_i \log \sum_\alpha q_\alpha P(D_i|\Lambda_\alpha)\,, \qquad (3.23)$$

which due to the summation within the log function would be much more difficult to maximize.

After each maximization (M-step) we set $\Lambda_n = \Lambda$, and take a next step, until convergence. With this choice of Λ_n (standard for the EM algorithm) the Jensen bound in line 3 becomes an equality, and the bound ensures that in the subsequent maximization step $\log P(D|\Lambda)$ will never decrease.

Appendix 3.C The artificial data set

The following tables present the parameters that are used to generate the artificial data in Section 3.5.1. Note that W and σ do not vary between clusters.

Table 3.3: Numerical values for the parameters generating the linear data.

	cluster 1	cluster 2
W	$\begin{pmatrix} 0.023 & -1.24 & -0.041 \end{pmatrix}$	
$\mathbf{m}_\alpha$	$\begin{pmatrix} -3.67 & 3.63 \end{pmatrix}$	$\begin{pmatrix} 0.048 & -2.43 \end{pmatrix}$
Σ_α	$\begin{pmatrix} 0.80 & 0.23 \\ 0.23 & 1.19 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.23 \\ -0.23 & 1.07 \end{pmatrix}$
σ	5	

Table 3.4: Numerical values for the parameters generating the nonlinear data.

	cluster 1	cluster 2
W	$\begin{pmatrix} 0.033 & -0.62 & -0.051 \end{pmatrix}$	
$\mathbf{m}_\alpha$	$\begin{pmatrix} -3.67 & 3.63 \end{pmatrix}$	$\begin{pmatrix} 0.048 & -2.43 \end{pmatrix}$
Σ_α	$\begin{pmatrix} 0.80 & 0.23 \\ 0.23 & 1.19 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.23 \\ -0.23 & 1.07 \end{pmatrix}$
W_0	0.27	
σ	5	

Chapter 4

The dynamic hierarchical model with time dependencies at lower levels

Abstract. We present a method to model parallel time series. We build upon the standard dynamic hierarchical model (DHM) as described e.g. in the work of Gamerman and Migon [31]. In this standard model the latent states that correspond to the parallel time series are all linked to one 'top-level' or 'average' dynamic linear model (DLM). The 'lower-level' states that describe the parallel time series are independent given the top-level states: there is no direct dependence between subsequent lower-level states. In this chapter we add these dependencies. Since exact inference becomes infeasible for large numbers of parallel time series, we propose two approximations, one a variational method, the other a factorial approximation. We apply the model to two data sets, one containing artificial data, the other real-world data. We obtain maximum likelihood (ML) values for the model parameters, and we apply both approximations and the exact model (where possible) to perform inference. We show that our method yields significant improvements to the standard model in terms of forecasting.

Adapted from: B. Bakker and T. Heskes. The dynamic hierarchical model with time dependencies at lower levels. *Journal of Time Series Analysis*, Submitted.

4.1 Introduction

Many real-world tasks can be viewed as parallel time series. Take for example weather prediction for various parts of the same country, stock price prediction for a portfolio of stocks traded on the same stock exchange, or sales figures for a number of different items sold in the same store. Models that describe such tasks can make use both of the fact that the data has the form of a time series, and therefore may have a specific behavior through time, and of the fact that these tasks are similar to each other, and thus may have a (hidden) inter-dependence.

In this chapter we will build upon the standard dynamic hierarchical model as presented in [31], which provides a general framework for modeling parallel time series. The standard model combines the hierarchical models of [57] and the dynamic linear models of [35]. It features a top-level (average) time series, modeled through a dynamic linear model (see e.g. [35, 86]). The latent state $\mathbf{M}_t$ of this series evolves through time, in a manner determined by an evolution matrix G_t. At each time t, the probability of lower-level states $\theta_{i,t}$, corresponding to the parallel time series, are inferred from the top-level state at the same time. The states of the parallel DLMs themselves do however not depend directly on previous states $\theta_{i,t-1}$. One lower-level state may therefore have one deviation from the top-level state at time t, yet a totally different, uncorrelated deviation at time $t+1$. In this chapter we extend the standard dynamic hierarchical model through addition of a time dependence within parallel series. In [31] it is argued that time evolution (dependencies between states at subsequent times) can occur only at the highest level of hierarchy. Although this is true when evolution only takes place at *one* level, we will show that inference is possible when *all* levels are linked through time (more specifically: as long as states at the highest level are linked through time, lower levels may be linked as well).

The addition of these dependencies, however, makes inference infeasible for large numbers of parallel series. We therefore present two approximating methods to perform inference on the proposed model. The first approximation makes use of the so-called variational approach (see also [47]): we construct an approximating model which consists of a single, independent DLM for each parallel time series, and one independent top-level DLM. This approximating model is optimized through minimization of its Kullback-Leibler (KL) divergence to the exact model. The second approximation is closely related to a local optimization method introduced in [12], where the approximating model consists of independent probability distributions for each individual state, including the top-level states.

We present our extension of the hierarchical model of [31] in Section 4.2. In Sections 4.3 and 4.4 we describe the aforementioned approximate inference methods. Related work and the extension of our model to higher-level hier-

archical models are discussed in Sections 4.5 and 4.6. We make use of exact inference, the two approximating methods and the standard DHM to find maximum likelihood values for the model parameters and to perform forecasting on two databases in Section 4.7. Section 4.8 concludes the chapter with a summary and an outlook on future work.

4.2 A hierarchical time series model

We consider a collection of n parallel time series indexed by i, each characterized by T combinations of a covariate $\mathbf{x}_{i,t}$ and a response $y_{i,t}$. The response is modeled as a linear function of corresponding covariates $\mathbf{x}_{i,t}$ with additional Gaussian noise $\epsilon_{i,t}$:

$$y_{i,t} = \boldsymbol{\theta}_{i,t}^T \mathbf{x}_{i,t} + \epsilon_{i,t} \,, \tag{4.1}$$

where we assume all noise terms $\epsilon_{i,t}$ to be independent of each other, and normally distributed around zero, with variance σ^2. $\boldsymbol{\theta}_{i,t}$ is the (time-dependent) regression parameter. It plays the role of a dynamic latent variable. In the standard model the lower-level states only depend on $\mathbf{M}_t$:

$$\boldsymbol{\theta}_{i,t} = F_{i,t}\mathbf{M}_t + \boldsymbol{\xi}_{i,t} \,, \tag{4.2}$$

where the noise $\boldsymbol{\xi}_{i,t}$ is assumed to be normally distributed around zero, with variance Σ. The evolution of the top-level states obeys

$$\mathbf{M}_t = G_t\mathbf{M}_{t-1} + \gamma_t \quad \text{with} \quad E\left[\gamma_t\gamma_t^T\right] = \Sigma_M \,. \tag{4.3}$$

We extend this model, which so far has followed the description outlined in [31], by changing Equation 4.2 to

$$\boldsymbol{\theta}_{i,t} = A_t\boldsymbol{\theta}_{i,t-1} + (\mathbb{1} - A_t)\mathbf{M}_t + \boldsymbol{\xi}_{i,t} \,. \tag{4.4}$$

The prediction for each new state is a weighted average of the old state, propagated through the evolution matrix A_t, and the corresponding top-level state. The top-level state thus couples the dynamics of the lower-level time series. Equation 4.4 can also be written in the form

$$\boldsymbol{\theta}_{i,t} = \mathbf{M}_t + A_t(\boldsymbol{\theta}_{i,t-1} - \mathbf{M}_t) + \boldsymbol{\xi}_{i,t} \,, \tag{4.5}$$

indicating that we are in fact modeling the dynamics of the deviations of the lower-level states to the top-level states. Initial conditions are:

$$\mathbf{M}_1 \sim \mathcal{N}(\hat{\mathbf{M}}_1, \Sigma_{M_1}) \quad \text{and} \quad \boldsymbol{\theta}_{i,1} \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_1, \hat{\Sigma}_1) \,. \tag{4.6}$$

In this thesis we will consider static evolution matrices, $A_t = A$ and $G_t = G$.

The dependence on the hierarchical state $\mathbf{M}_t$ in Equation 4.4 implements one form of parameter sharing between the lower-level DLMs. A further sharing of parameters can be implemented on the level of observations, through a common parameter W. The shared parameter W is incorporated in Equation 4.1, yielding

$$y_{i,t} = \boldsymbol{\theta}_{i,t}^T W \mathbf{x}_{i,t} + \epsilon_{i,t} \,, \tag{4.7}$$

with W an $n_{\text{dim}} \times n_{\text{input}}$ dimensional matrix that can be learned from the data. When n_{dim} is smaller than n_{input} this matrix implements a lower-dimensional representation of the covariates. A similar sharing of one lower-dimensional representation can be found in the static hierarchical model proposed in Chapter 3 of this thesis. Implementation of this lower-dimensional representation is easy, and does not influence the presentation of our methods.

The model is visualized graphically in Figure 4.1. A *directed graphical model* like this represents each latent state by a separate node (ellipse). Lines between nodes represent conditional independencies. Unconnected nodes are said to be conditionally independent. Take for example nodes $\boldsymbol{\theta}_{2,t-1}$, $\boldsymbol{\theta}_{2,t}$ and $y_{2,t}$. Node $y_{2,t}$ is connected to $\boldsymbol{\theta}_{2,t}$, but not to $\boldsymbol{\theta}_{2,t-1}$. Therefore we can say that $y_{2,t}$ is conditionally independent of $\boldsymbol{\theta}_{2,t-1}$ given $\boldsymbol{\theta}_{2,t}$, or $P(y_{2,t}|\boldsymbol{\theta}_{2,t-1}, \boldsymbol{\theta}_{2,t}) = P(y_{2,t}|\boldsymbol{\theta}_{2,t})$. More information on graphical models can be found e.g. in [67].

4.2.1 Optimization

We implement optimization through a maximum likelihood approach. That is, we search for the parameters that optimize $\log P(Y_{1..T}|\Lambda, X_{1..T})$, the (log) probability of *all* observations $Y_{1..T}$, given the model parameters

$$\Lambda = \{\Sigma, \Sigma_M, \sigma, A, G, \hat{\mathbf{M}}_1, \Sigma_{M_1}, \hat{\boldsymbol{\theta}}_1, \hat{\Sigma}_1, (W)\} \,, \tag{4.8}$$

and all covariates $X_{1..T}$. We will assume from here on that $X_{1..T}$ is fixed and given, and omit it from our notation. Further, we define a super state

$$\mathbf{Z}_t = [\mathbf{M}_{t+1}, \boldsymbol{\theta}_{1t}, \ldots, \boldsymbol{\theta}_{nt}] \,, \tag{4.9}$$

and denote by $Z_{1..T}$ all latent states in the model up to time T.

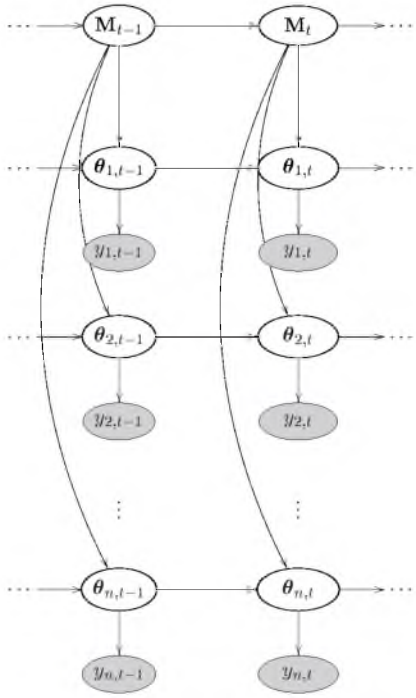


Figure 4.1: Graphical model. The shaded areas represent the observations $y_{i,t}$, the open ellipses represent the latent states. The top-level states (upper ellipses) are connected to all of the lower-level states (lower ellipses). Covariates $\mathbf{x}_{it}$ are left out for clarity.

For the optimization task we propose an expectation-maximization (EM) algorithm (see e.g. [75]). The EM algorithm is an iteration of two steps:

- E-step: we calculate $\langle \log P(Z_{1..T}, Y_{1..T}|\Lambda) \rangle_{P(Z_{1..T}|Y_{1..T},\Lambda_m)}$, the expectation value of the log probability of the latent states and the observations given Λ_m. The set of model parameters Λ_m is initialized (as Λ_0) at random at the start of the algorithm.
- M-step: we maximize the expression for $\langle \log P(Z_{1..T}, Y_{1..T}|\Lambda) \rangle_{P(Z_{1..T}|Y_{1..T},\Lambda_m)}$ with respect to Λ to obtain the new Λ_{m+1}.

These two steps are iterated until convergence, when Λ_{m+1} is no longer significantly different from Λ_m. The EM algorithm is guaranteed to converge to a (local) maximum of $\log P(Y_{1..T}|\Lambda)$. The expression that is maximized in the M-step is a quadratic expression in Λ, which makes this step relatively easy. The more involved part of the algorithm is the inference in the E-step. In 4.2.2 we give expressions for the required probability distributions, and in Sections 4.3 and 4.4 we describe approximations to perform this inference in linear time.

4.2.2 Inference

The optimization in 4.2.1 requires the calculation of the joint probability $P(Z_{1..T}, Y_{1..T}|\Lambda)$. The marginal probability of the latent states $Z_{1..T}$ and $P(Z_{1..T}|Y_{1..T}, \Lambda)$ is directly related to the joint probability:

$$P(Z_{1..T}|Y_{1..T}) = \frac{P(Z_{1..T}, Y_{1..T})}{P(Y_{1..T})} \,. \tag{4.10}$$

The joint probability itself can be written as a product of "two-slice potentials"

$$P(Z_{1..T}, Y_{1..T}) = \prod_t \Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t) \,, \tag{4.11}$$

with, for the hierarchical model of Figure 4.1,

$$\begin{aligned} \Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t) &= P(Y_t|\mathbf{Z}_t) P(\mathbf{Z}_t|\mathbf{Z}_{t-1}) \\ &= P(Y_t|\Theta_t) P(\Theta_t|\Theta_{t-1}, \mathbf{M}_t) P(\mathbf{M}_t|\mathbf{M}_{t-1}) \,, \end{aligned} \tag{4.12}$$

where we can further decompose

$$P(Y_t|\Theta_t) = \prod_i P(y_{i,t}|\boldsymbol{\theta}_{i,t}) \quad \text{and} \quad P(\Theta_t|\Theta_{t-1}, \mathbf{M}_t) = \prod_i P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) \,. \tag{4.13}$$

The potential on the boundary $t = 1$ is slightly different:

$$\Psi_1(\mathbf{Z}_0, \mathbf{Z}_1) = \Psi_1(\mathbf{Z}_1) = P(Y_1|\Theta_1)P(\Theta_1)\,, \tag{4.14}$$

where $P(\Theta_1)$ is defined through Equation 4.6. Exactly the same ideas apply to filtering and forecasting, the latter of which is implemented in Section 4.7.

If we look at our dynamic hierarchical model in terms of the super state $\mathbf{Z}_t$ defined in 4.2.1, we can express the model as one large DLM and, in principle, we can apply the standard procedures for forecasting, filtering and smoothing. The evolution equation for $\mathbf{Z}_t$ reads

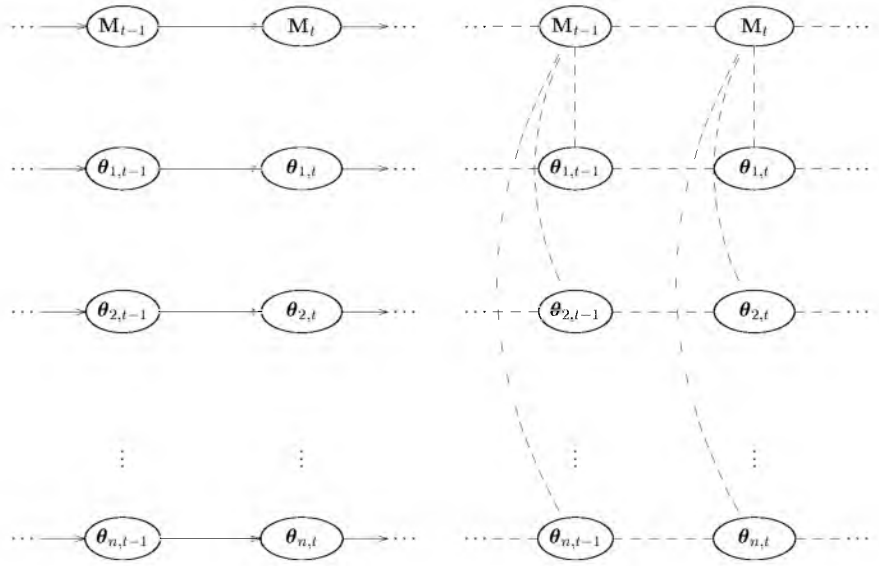
$$\begin{pmatrix} \mathbf{M}_{t+1} \\ \boldsymbol{\theta}_{1,t} \\ \boldsymbol{\theta}_{2,t} \\ \vdots \\ \boldsymbol{\theta}_{n,t} \end{pmatrix} = \begin{pmatrix} G & \mathbb{0} & \cdots & \cdots & \mathbb{0} \\ \mathbb{1} - A & A & \mathbb{0} & & \vdots \\ \mathbb{1} - A & \mathbb{0} & A & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{0} \\ \mathbb{1} - A & \mathbb{0} & \cdots & \mathbb{0} & A \end{pmatrix} \begin{pmatrix} \mathbf{M}_t \\ \boldsymbol{\theta}_{1,t-1} \\ \boldsymbol{\theta}_{2,t-1} \\ \vdots \\ \boldsymbol{\theta}_{n,t-1} \end{pmatrix} + \Xi_t \tag{4.15}$$

where the noise covariance matrix Γ is of the form

$$E\left[\Xi_t \Xi_t^T\right] = \begin{pmatrix} \Sigma_M & \mathbb{0} & \cdots & \mathbb{0} \\ \vdots & \Sigma & & \mathbb{0} \\ \mathbb{0} & & \ddots & \vdots \\ \mathbb{0} & \cdots & \mathbb{0} & \Sigma \end{pmatrix}. \tag{4.16}$$

Clearly, inference is possible for a dynamic hierarchical model where both the top-level series $\mathbf{M}_t$ and the lower-level series $\boldsymbol{\theta}_{it}$ are linked through time. Note however that, if only the lower-level states would be thus linked (i.e. the top-level DLM would not feature dependencies between states at subsequent times), the above description would fail. The reason for this failure would be that the propagation term would feature no prediction for $\mathbf{M}_{t+1}$, which can therefore not be part of the state vector $\mathbf{Z}_t$. More generally: when neither $P(\mathbf{M}_{t+1}|\mathbf{M}_t)$ nor $P(\mathbf{M}_t|\Theta_t)$ is defined, $\mathbf{M}_t$ cannot be inferred from the data (see also [31]). This is why a dynamic hierarchical model of any order must always have time evolution on the highest level. The lower-level states, which can be inferred from the top-level states, do not *need* time evolution. There is however no reason to forbid time evolution on the lower level, this just yields some extra terms in the evolution matrix. (Note that none of this is in actual disagreement with the statements in [31], we just emphasize a different aspect.)

It is theoretically possible to perform exact inference on Z. However, the methods involved in this procedure require the inversion of matrices of size $(n+1)n_{\text{dim}} \times (n+1)n_{\text{dim}}$, with n the number of parallel series and n_{dim} the



(a) Graphical structure of the approximate model used in the variational approximation

(b) Graphical structure of the approximate model used in the factorial approximation

Figure 4.2: Graphical structure of the approximate models used in the variational approximation (left) and the factorial approximation (right). Ellipses represent latent states $\mathbf{M}_t$ for the top-level DLM and $\boldsymbol{\theta}_{i,t}$ for the lower-level time DLMs. Observations are left out for clarity. The dashed lines in the right graph indicate that although the approximate model is fully factorized, the connections between states are incorporated in each iteration step.

dimension of the latent states. This becomes practically infeasible for large n, which is why in Sections 4.3 and 4.4 we will introduce two methods for approximate inference that scale linearly with the number of parallel series.

4.3 Variational approximation

In the exact posterior $P(Z_{1..T}|Y_{1..T})$ all lower-level states are coupled to top-level states, and within each DLM states at subsequent times are linked. The variational approximation 'cuts' the dependencies between the top-level states

$\mathbf{M}_t$ and the lower-level states $\boldsymbol{\theta}_{i,t}$ (see Figure 4.2a). Following e.g. [32, 47], we approximate the exact posterior with a distribution of the form

$$P(Z_{1..T}|Y_{1..T}) \approx Q(Z_{1..T}) = Q_0(\mathbf{M}_{1..T}) \prod_i Q_i(\boldsymbol{\theta}_{i,1..T}) = \prod_{i=0}^{n} Q_i(\boldsymbol{\theta}_{i,1..T}) \,, \tag{4.17}$$

where for notational convenience we defined $\boldsymbol{\theta}_{0,t} = \mathbf{M}_t$. This distribution uncouples the dynamics of the lower-level DLMs and the top-level DLM but can still take into account time dependencies. In the variational approach, the 'best' approximation $Q(Z_{1..T})$ is defined to be the one that minimizes the KL-divergence

$$\mathrm{KL}[Q,P] = \int dZ_{1..T} Q(Z_{1..T}) \left[\log Q(Z_{1..T}) - \log P(Z_{1..T}|Y_{1..T})\right] \,. \tag{4.18}$$

When we optimize $\mathrm{KL}[Q,P]$ for $Q_i(\boldsymbol{\theta}_{i,1..T})$, keeping all others fixed, the optimal $Q_i(\boldsymbol{\theta}_{i.1..T})$ is easily found to obey (see Appendix 4.A)

$$Q_i(\boldsymbol{\theta}_{i,1..T}) \propto \exp \langle \log P(Z_{1..T}, Y_{1..T}) \rangle_{Q_{-i}} \,, \tag{4.19}$$

where $\langle .. \rangle_{Q_{-i}}$ denotes the average over the distribution

$$Q_{-i}(Z_{1..T}) = \prod_{j=0;j\neq i}^{n} Q_j(\boldsymbol{\theta}_{j,1..T}) \,. \tag{4.20}$$

The standard procedure is to iterate these so-called 'mean-field' equations for each $Q_i(\boldsymbol{\theta}_{i.1..T})$ until convergence.

4.3.1 The hierarchical model

The mean-field equations for the two-level hierarchical model take the following form. Let us first consider optimizing $Q_i(\boldsymbol{\theta}_{i.1..T})$ for one of the lower-level series ($i \neq 0$). Substituting (4.12) and (4.13), we obtain

$$Q_i(\boldsymbol{\theta}_{i,1..T}) \propto \prod_t P(y_{i,t}|\boldsymbol{\theta}_{i,t}) \exp \langle \log P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) \rangle_{Q_0} \,. \tag{4.21}$$

This can be interpreted as the posterior of a standard Markov model with transformed states $\tilde{\boldsymbol{\theta}}_{it}$ and observations $\tilde{y}_{it}$. A detailed description is presented in Appendix 4.A.

Similarly, fixing all $Q_i(\boldsymbol{\theta}_{i,1..T})$, the optimal $Q_0(\mathbf{M}_{1..T})$ can be found to obey

$$Q_0(\mathbf{M}_{1..T}) \propto \prod_t P(\mathbf{M}_t|\mathbf{M}_{t-1}) \prod_i \exp \langle \log P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) \rangle_{Q_i} \,. \tag{4.22}$$

This can also be interpreted as the posterior of a Markov model if we can define an observation equation $P(\tilde{y}_t|\mathbf{M}_t)$ and observations $\tilde{y}_t$ such that

$$P(\tilde{y}_t|\mathbf{M}_t) \propto \prod_i \exp\left\langle \log P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t)\right\rangle_{Q_i} . \tag{4.23}$$

$P(\tilde{y}_t|\mathbf{M}_t)$ and $\tilde{y}_t$ are defined in Appendix 4.A.

The $Q_i(\boldsymbol{\theta}_{i.1..T})$ depend on $Q_0(\mathbf{M}_{1..T})$ only through the expectation values $\langle \mathbf{M}_t \rangle$, and vice versa. This leads to the following iterative procedure:

1. Start with random values for $\langle \mathbf{M}_t \rangle$.
2. Construct the posterior of the lower-level states given $\langle \mathbf{M}_t \rangle$.
3. Infer $\langle \boldsymbol{\theta}_{i,t} \rangle$ from this posterior, for $i = 1..n$.
4. Construct the posterior of the top-level states given $\langle \boldsymbol{\theta}_{i,t} \rangle$.
5. Infer $\langle \mathbf{M}_t \rangle$ from this posterior.
6. Repeat from step 2 until convergence.

In each iteration step we perform inference on $n + 1$ independent DLMs: one for the top-level states and n for the lower-level states. The approximate distribution obtained in this way is a first-order Markov model, and is visualized in Figure 4.2a.

It can be shown (see Appendix 4.B) that the marginal means $\langle \boldsymbol{\theta}_{i,t} \rangle$, $\langle \mathbf{M}_t \rangle$ under the (converged) approximating distribution are identical to the marginal means under the exact distribution. Any discrepancy between approximate and exact inference is therefore due to the error made in the estimation of the variances.

4.4 Factorial approach

As an alternative to the variational approximation, we consider an iterative variant of the Boyen-Koller algorithm [12] known as 'expectation propagation' (see [42, 63]). We approximate the posterior $P(Z_{1..T}|Y_{1..T})$ with a distribution of the form

$$Q(Z_{1..T}) = \prod_t Q_t(\mathbf{Z}_t) = \prod_t Q_{0,t}(\mathbf{M}_t) \prod_i Q_{i,t}(\boldsymbol{\theta}_{i,t}) . \tag{4.24}$$

Note that this distribution is fully factorized, i.e. does not contain any links between consecutive states. We further decompose Q into

$$Q(Z_{1..T}) = \prod_t \lambda_t(\mathbf{Z}_t)\mu_t(\mathbf{Z}_t) \quad \text{with} \quad \lambda_t(\mathbf{Z}_t) = \lambda_{0,t}(\mathbf{M}_t) \prod_i \lambda_{i,t}(\boldsymbol{\theta}_{i,t}) , \tag{4.25}$$

and similarly for $\mu_t(\mathbf{Z}_t)$.

At first sight, this factorized distribution seems to be less powerful than the approximation in (4.17) for the variational approach. However, at each update we add a link in the form of a potential $\Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$, yielding an approximate distribution $Q_{t-1:t}(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$. Taking into account these two-slice marginals, we do implicitly have an approximation corresponding to the first-order Markov model visualized in Figure 4.2b.

We incorporate the potential $\Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$ in our approximation by rewriting $Q(Z_{1..T})$ to

$$Q(Z_{1..T}) \propto \prod_{t'<t-1} Q_{t'}(\mathbf{Z}_{t'}) \lambda_{t-1}(\mathbf{Z}_{t-1}) \Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t) \mu_t(\mathbf{Z}_t) \prod_{t'>t} Q_{t'}(\mathbf{Z}_{t'}) \,; \quad (4.26)$$

this shows how $\Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$ forms the 'link' between the states at times t and $t-1$. In this form the joint distribution of $\mathbf{Z}_{t-1}$ and $\mathbf{Z}_t$ reads

$$Q_{t-1:t}(\mathbf{Z}_{t-1}, \mathbf{Z}_t) = \frac{1}{c_t} \lambda_{t-1}(\mathbf{Z}_{t-1}) \Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t) \mu_t(\mathbf{Z}_t) \,, \quad (4.27)$$

with c_t a normalization constant. These normalization constants can be used to estimate the data likelihood:

$$P(Y_{1..T}) \approx \prod_{t=1}^{T} c_t \,. \quad (4.28)$$

The factorial approach can be called 'greedy', since it iterates over local optimization steps instead of performing one global optimization. Each iteration step involves an update, where (part of) the factorial distribution is fitted to Equation 4.27. In the update procedure, we can distinguish two steps: forward and backward.

Forward pass to update $\lambda_t(\mathbf{Z}_t)$. We compute $Q_t(\mathbf{Z}_t)$ by integrating out $\mathbf{Z}_{t-1}$ from $Q_{t-1:t}(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$. Since $Q_t(\mathbf{Z}_t)$ itself does not factorize, we project back to a factorized distribution through marginalization, yielding $Q_{0,t}(\mathbf{M}_t)$ and $Q_{i,t}(\boldsymbol{\theta}_{i,t})$. The new term approximations follow from

$$\lambda_{0,t}(\mathbf{M}_t) = \frac{Q_{0,t}(\mathbf{M}_t)}{\mu_{0,t}(\mathbf{M}_t)} \quad \text{and} \quad \lambda_{i,t}(\boldsymbol{\theta}_{i,t}) = \frac{Q_{i,t}(\boldsymbol{\theta}_{i,t})}{\mu_{i,t}(\boldsymbol{\theta}_{i,t})} \,. \quad (4.29)$$

Backward pass to update $\mu_{t-1}(\mathbf{Z}_{t-1})$. This is a perfect mirror of the forward pass: compute $Q_{t-1}(\mathbf{Z}_{t-1})$, $Q_{0,t-1}(\mathbf{M}_{t-1})$, and $Q_{i,t-1}(\boldsymbol{\theta}_{i,t-1})$ through marginalization and update the term approximations according to

$$\mu_{0.t-1}(\mathbf{M}_{t-1}) = \frac{Q_{0,t-1}(\mathbf{M}_{t-1})}{\lambda_{0,t-1}(\mathbf{M}_{t-1})} \quad \text{and} \quad \mu_{i,t-1}(\boldsymbol{\theta}_{i,t-1}) = \frac{Q_{i,t-1}(\boldsymbol{\theta}_{i,t-1})}{\lambda_{i,t-1}(\boldsymbol{\theta}_{i,t-1})} \,. \quad (4.30)$$

In principle the choice of the ordering of the updates is free. However, it makes sense to iterate going forward and backward in time: first compute all λ's from $t = 1$ to T, then all μ's from $t = T$ to 1, then update all λ's from $t = 1$ to T, and so forth until convergence.

The practice of calculating the exact marginal (here $Q_t(\mathbf{Z}_t)$) and then projecting this marginal back to a simpler approximating distribution ($Q_{i.t}(\boldsymbol{\theta}_{i,t})$) is used in many other inference methods as well. Examples are Bayesian forecasting [35], the extended Kalman filter [49] and generalized pseudo-Bayes for switching linear dynamical systems [42, 53].

At convergence, we can improve upon the fully factorized approximation by computing the two-slice distributions

$$Q_{0,t-1:t}(\mathbf{M}_{t-1}, \mathbf{M}_t) = \int d\boldsymbol{\theta}_{1,t-1} \dots d\boldsymbol{\theta}_{n,t-1} d\boldsymbol{\theta}_{1,t} \dots d\boldsymbol{\theta}_{n,t} Q_{t-1:t}(\mathbf{Z}_{t-1}, \mathbf{Z}_t) \,, \tag{4.31}$$

and similarly for $Q_{i,t-1:t}(\boldsymbol{\theta}_{i,t-1}, \boldsymbol{\theta}_{i,t})$. By construction, the single-slice marginals $Q_{0,t-1}(\mathbf{M}_{t-1}), Q_{0,t}(\mathbf{M}_t), Q_{i,t-1}(\boldsymbol{\theta}_{i,t-1})$ and $Q_{i,t}(\boldsymbol{\theta}_{i,t})$ are consistent with these two-slice marginals.

The factorial approximation features exact means as well. This is shown in Appendix 4.C. Another way to show that this approximation features exact means is presented in [77].

4.5 Generalization to more than two levels

It is straightforward to generalize the exact hierarchical method and the two approximations to more than two levels. We outline the extension to a three-level model (see also Figure 4.3). Addition of subsequent higher levels proceeds in the same way.

Parameters at the lowest level, which refer to the lower-level series and couple directly to the observations, are denoted $\boldsymbol{\psi}_{i,j,t}$. These parameters represent latent states that, as before, depend on the previous states $\boldsymbol{\psi}_{i,j,t-1}$ and on a mid-level state, denoted $\boldsymbol{\theta}_{i,t}$. These mid-level states $\boldsymbol{\theta}_{i,t}$ are part of an ensemble of 'mid-level DLMs', that constitute the second level. Each state on this level is connected to a selection from the lowest-level states (all with the same index i), and to the top-level states $\mathbf{M}_t$.

In the variational approximation we now iterate over three levels. The approximations for the two outer levels are expressed as before, for the two-level model. The solution for the middle level contains averages over both lowest- and mid-level dynamics:

$$Q_i(\boldsymbol{\theta}_{i,1..T}) \propto \prod_t \exp \left\langle \log P(\boldsymbol{\psi}_{i,j,t} | \boldsymbol{\psi}_{i,j,t-1}, \boldsymbol{\theta}_{i,t}) \right\rangle_{Q_{i,j}} \exp \left\langle \log P(\boldsymbol{\theta}_{i,t} | \boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) \right\rangle_{Q_0} . \tag{4.32}$$

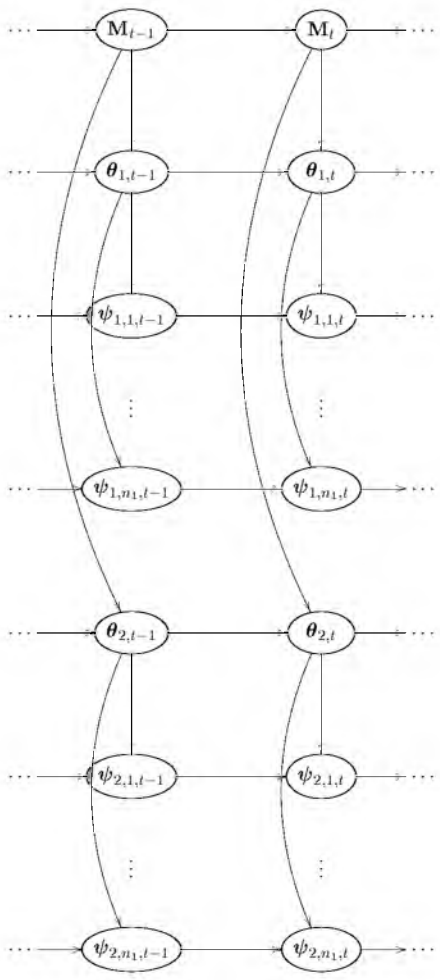


Figure 4.3: Three-level model. The top ellipses represent the latent states of the top-level states $\mathbf{M}_t$. The latent states $\boldsymbol{\theta}_{i,t}$, which are connected to $\mathbf{M}_t$, are on the second level. Each of these states is connected to a set of latent states $\boldsymbol{\psi}_{i,j,t}$ on the lowest level. Observations (left out for clarity) are connected to the lowest-level states.

In the corresponding mean field equations, the average over the top-level introduces a bias term with extra noise and the lower level introduces observations. The obvious iteration scheme is lower-middle-top-middle-lower-middle-top and so on.

The factorial approach proceeds in much the same way as before. The two-slice potential $\Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t)$ now contains terms from three levels. These terms include connections within each level as well as connections between

lowest- and mid-level states, and between mid- and top-level states. For the three-level model of Figure 4.3 this potential reads

$$\begin{aligned}\Psi_t(\mathbf{Z}_{t-1}, \mathbf{Z}_t) &= P(Y_t|\mathbf{Z}_t)P(\mathbf{Z}_t|\mathbf{Z}_{t-1}) \\ &= P(Y_t|\Psi_t)P(\Psi_t|\Psi_{t-1}, \Theta_t)P(\Theta_t|\Theta_{t-1}, \mathbf{M}_t)P(\mathbf{M}_t|\mathbf{M}_{t-1}) ,\end{aligned} \tag{4.33}$$

where we can further decompose

$$P(Y_t|\Theta_t) = \prod_i P(y_{i,t}|\boldsymbol{\theta}_{i,t}), \tag{4.34}$$

$$P(\Psi_t|\Psi_{t-1}, \Theta_t) = \prod_i \prod_{j=1}^{n_i} P(\boldsymbol{\psi}_{i,j,t}|\boldsymbol{\psi}_{i,j,t-1}, \boldsymbol{\theta}_{i,t}) \quad \text{and} \tag{4.35}$$

$$P(\Theta_t|\Theta_{t-1}, \mathbf{M}_t) = \prod_i P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) , \tag{4.36}$$

where n_i is the number of lower-level DLMs that is connected to the mid-level DLM $\boldsymbol{\theta}_{i,t}$. Forward and backward passes are performed in the same way as before, where marginalization now yields factorized distributions $Q_{0,t}(\mathbf{M}_t)$, $Q_{i,j,t}(\boldsymbol{\psi}_{i,j,t})$ and $Q_{i,t}(\boldsymbol{\theta}_{i,t})$.

4.6 Related work

Although in this chapter we have concentrated on the work of Gamerman and Migon [31], a wider literature exists on the subject of hierarchical dynamic models. Cargnoni *et al* [22] presented a model with the same hierarchical structure as [31], but for non-normal multivariate time series. The non-normality of this model prevents exact inference, and sampling methods are proposed to simulate the posterior.

An alternative method to modeling parallel time series has been presented by Zhang *et al* [88]. Each parallel time series is modeled as the sum of a common (for all time series) smoothing spline function, a regression term (or fixed effect) $\mathbf{x}_{i,t}^T\beta$ on the covariates $\mathbf{x}_{i,t}$ and a series-dependent random effects term $\mathbf{z}_{i,t}^T\mathbf{b}_i$, where $\mathbf{z}$ is the subset of the covariates that corresponds to the random effect.

More work on time series modeling through the use of random effects has been done by Aguilar and West [1]. Here, the fixed effect is modeled through a DLM structure, whereas the random effects are not linked through time.

Camargo and Gamerman [21] have combined a hierarchical model structure with a DLM structure within parallel time series through a mixture model. The mixture elements are the hierarchical model of [31], and independent DLMs for each parallel series. The probability of the mixture components is a time-dependent Bernoulli distribution.

An example of a more traditional approach can be found in the work of Bunn and Vassilopoulos [19], who implement a combination of individual and group seasonal estimation for short-term sales forecasting of multiple retail products.

The work presented in this chapter is to our knowledge the first implementation of a graphical model that features dependencies both between lower-level states at subsequent times and between top-level and lower-level states.

4.7 Results

Description of the data. We tested our model on two databases, one created artificially, the other containing real-world data. The covariates $\mathbf{x}_{i,t}$ and the initial state means $\boldsymbol{\theta}_1, \mathbf{M}_1$ in the first database were drawn from a normal distribution with zero mean and unit variance. Propagation matrices for the lower-level DLMs and the top-level DLM and covariance matrices for the three sources of noise (on the top-level state predictions, on the lower-level state predictions and on the outputs) were also chosen at random. The latent states were created to be 3-dimensional. Given these parameters, outputs $y_{i,t}$ were generated according to Equations 4.1, 4.3 and 4.4.

The second database contains sales figures for single copy newspaper sales in the Netherlands. The outputs represent the numbers of newspapers sold on 156 consecutive Saturdays, at 343 separate outlets throughout the Netherlands. Inputs contain information about the weather, season, short-term previous sales (4 to 6 weeks ago) and long-term previous sales (54 to 56 weeks ago). Since in this case the dimension of the inputs is rather high (13), we implemented a lower-dimensional representation through the addition of a 3×13 matrix W (see Section 4.2).

Quality of the approximations. For both the top-level series and the lower-level series we inferred the single state marginals $P(\boldsymbol{\theta}_{i,t}|Y_{1..T}, \Lambda)$ under the exact posterior and both the variational and the factorial approximation. We rated the quality of both approximations through the KL-divergences between the approximated and the exact marginals. Inference was performed on random selections of series, taken from the artificial data set. For each random draw we selected n (varying from 10 to 50) different series of 80 consecutive observations and covariates. The model parameters Λ were assigned randomly for each draw, and for each n we performed inference on 10 independent draws. The mean-dependent parts of the KL-distances for both approximations were always zero after convergence, as expected. The variance-dependent parts of the distances (averaged over the 10 independent draws) are shown in Figure 4.4, as a function of n. The factorial approximation is significantly more accurate than the variational approximation for any number of tasks, although the difference does get smaller for increasing n. An example of the difference

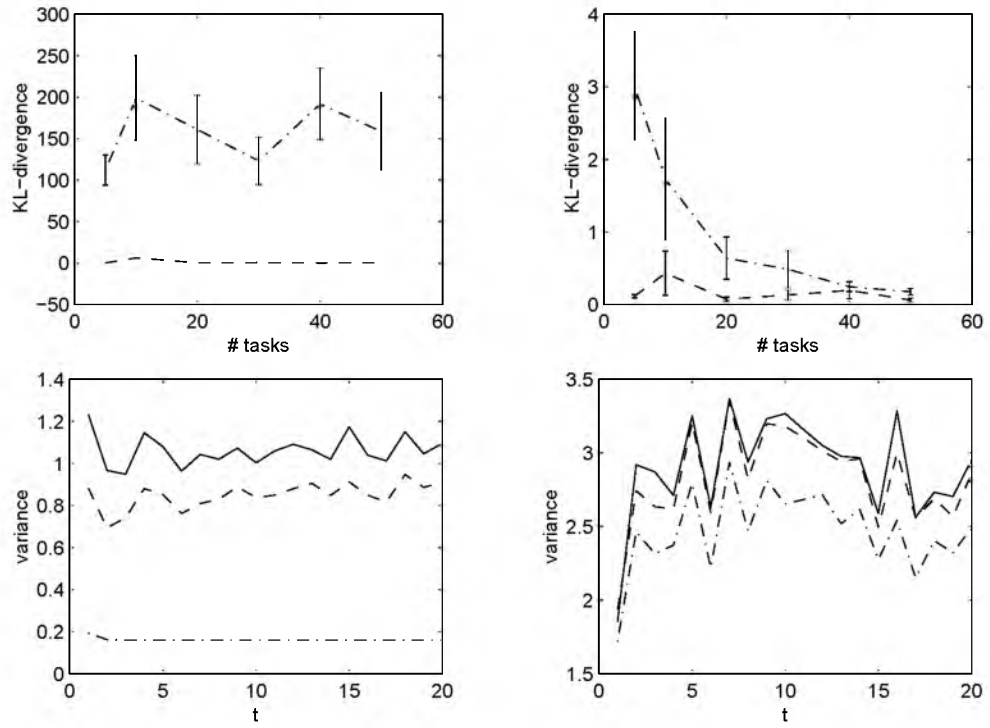


Figure 4.4: Top: Average variance-dependent parts of the KL-divergences between the approximated marginals and the exact marginals. The left panel plots the average divergences between marginals of the top-level series, the right panel plots the average divergences between the marginals of the lower-level series. The dash-dotted line plots the divergence between the variational approach and the exact model, the dashed line the divergence for the factorial approach. Bottom: the variance of the first element of the top-level state (left) and one from a set of 10 lower-level states (right) over time. Solid lines correspond to the exact model, dash-dotted lines to the variational approximation and dashed lines to the factorial approximation.

in estimated variance between the two approximations is given in the lower panels of the same figure.

Forecasting. For both databases we applied the EM algorithm to find ML model parameters for different numbers of parallel tasks (n). For the inference step in this algorithm we used either exact inference, or one of the approximating methods. For the newspaper database we also applied inference based on the standard DHM (without connections between subsequent lower-level states). For each number of tasks, and for each inference method, we performed 10 independent optimizations, where each time we took a ran-

dom selection of n tasks to be used for optimization and evaluation. The first 80 observation/covariate pairs of each task were used for optimization, the subsequent 20 data samples ($\mathbf{x}_{i,t}, y_{i,t}$ for $t = 81 \ldots 100$) were used to perform one-step-ahead forecasting. That is, we obtained ML parameters Λ_{ML}, and used them to calculate

$$\langle y_{i,t} \rangle_{P(y_{i,t}|\mathbf{x}_{i,t}, Y_{1..t-1}, \Lambda_{\text{ML}})} \ , \tag{4.37}$$

for $t = 81 \ldots 100$, starting with $t = 81$. After prediction of each $y_{i,t}$ we used the true value for $y_{i,t}$ to update the posterior, where we kept the old values for Λ_{ML}. In each trial we used the same inference method (exact or approximate) both for optimization and for posterior updates. Each of the trials was rated through average squared error

$$E = [20 \cdot n]^{-1} \sum_{i=1}^{n} \sum_{t=81}^{100} (\langle y_{i,t} \rangle - y_{i,t})^2 \ , \tag{4.38}$$

and computation time.

Figure 4.5 shows the average squared error and required computation time as a function of the number of parallel series for the artificial data set. Results are displayed for both approximation methods and for exact inference. It can be seen that, whereas the computation time for exact inference grows strongly with the number of parallel tasks, for either form of approximate inference it grows only linearly. This gain of speed infers but a small loss of accuracy: the average squared error of the approximations is not significantly higher than the error incurred through exact inference. Similar results were obtained from the newspaper data. For very small numbers of tasks, the results for exact inference are worse than for approximate inference. This is due to a form of 'overfitting': although the observations that were used for optimization have a high probability under the model with parameters Λ_{ML}, the model generalizes poorly for new observations. Note however that parallel time series modeling is aimed at larger numbers of parallel tasks, where this problem no longer occurs.

Performance on the newspaper data is presented in Figure 4.6. It can be seen that with only a few parallel tasks the average squared error drops strongly. No further improvements are gained for more than 6 parallel tasks. The optimization process appears not to be hindered by the use of approximate inference instead of exact inference. The standard DHM however, incurs a clearly higher error than all of the methods presented in this chapter.

Figure 4.7 shows the means over time for the first elements of the latent state vectors for 4 parallel newspaper outlets and the top-level DLM. Plots are drawn for exact inference and for the standard DHM. It can be seen that under the latter model the dynamics for the parallel series are less smooth than for

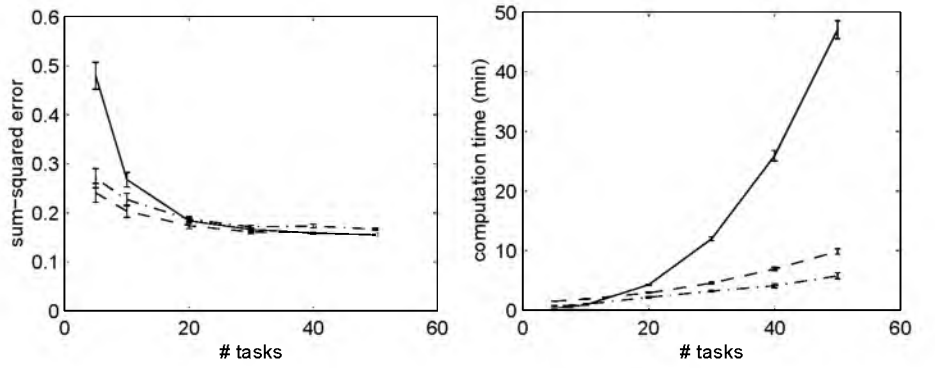


Figure 4.5: Average squared error (left) and computation time (right) as a function of the number of parallel tasks for the variational approximation (dash-dotted line), the factorial approximation (dashed line) and the exact model (solid line).

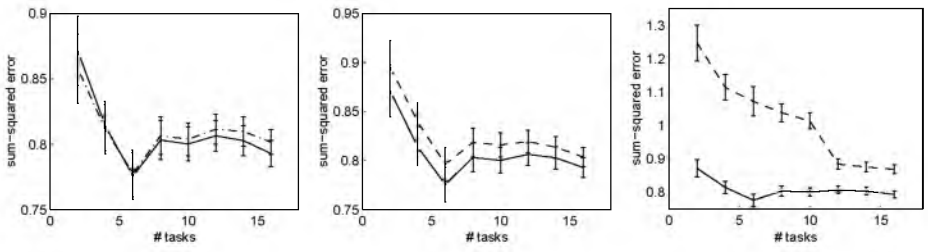


Figure 4.6: The average squared error on the newspaper data as a function of the number of parallel tasks for the variational approximation (dash-dotted line, left), the factorial approximation (dashed line, center) and the standard DHM (dashed line, right). The performance of the exact inference solution is drawn as a solid line in each panel.

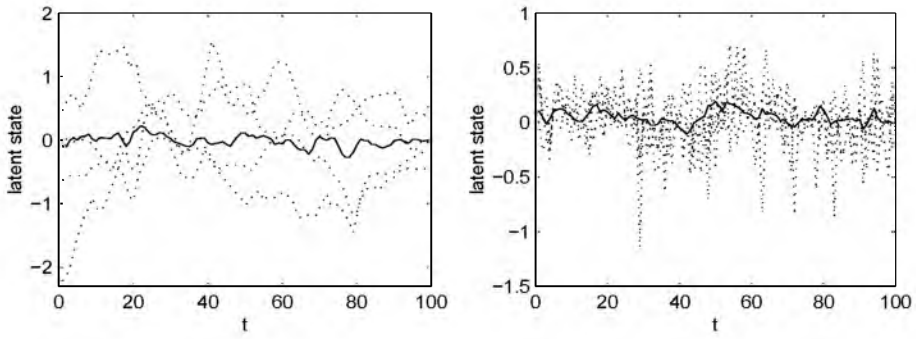


Figure 4.7: Means for the first elements of the latent states over time for 4 outlets from the newspaper data set, and the corresponding top-level DLM. Dotted lines correspond to lower-level states, solid lines represent top-level states. The left panel plots the exact means for the hierarchical model presented in this chapter, means inferred from the standard hierarchical model are plotted in the right panel.

our approach. This is due to the fact that lower-level states in the standard DHM are not linked through time.

Typical examples for the ML parameters A (for the lower-level dynamics) and G (for the top-level states), and the corresponding covariance matrices Σ and Σ_M are

$$A = \begin{pmatrix} 0.58 & 0.08 & -0.22 \\ -0.08 & 0.84 & 0.18 \\ -0.03 & 0.02 & 0.55 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 2.17 & 0.37 & 0.10 \\ 0.37 & 1.47 & -0.07 \\ 0.10 & -0.07 & 0.66 \end{pmatrix},$$

$$G = \begin{pmatrix} -0.10 & 0.13 & -0.03 \\ -0.00 & 0.30 & 0.10 \\ -0.03 & 0.02 & -0.22 \end{pmatrix}, \quad \Sigma_M = \begin{pmatrix} 0.58 & 0.10 & 0.01 \\ 0.10 & 0.61 & 0.02 \\ 0.01 & 0.02 & 0.42 \end{pmatrix}.$$

It can be seen that the optimized lower-level evolution matrix A has entries such that predictions for the next lower-level state depend both on the current lower-level state and on the top-level state.

4.8 Discussion

In this chapter we have extended the standard dynamic hierarchical model as described in [31]. Our extension introduced a time dependence between the lower-level states in this model, which proved beneficial on a database of real-world data: predictions based on the new model were more accurate than those based on the existing model.

Exact inference in the extended model is not practically feasible for large numbers of parallel time series. We therefore presented two methods for approximate inference and showed that both methods, although they are approximations, do feature exact means for the latent states. Comparison of the approximations to exact inference on two different databases confirmed that whereas for exact inference the required computation time rises strongly, for the approximating methods this increase is only linear. The performance after approximate inference was shown to be competitive with the performance after exact inference.

The two approximating methods were compared two each other, both in terms of KL-divergence between approximate and exact marginals and in terms of forecasting and required computation time. Inference through the slightly slower factorial approximation was shown to be closer to exact than inference through the variational approximation. This superiority with respect to inference did however not cash out in the form of better predictions (since both approximations did not perform worse than the exact method anyway). Nevertheless, other, more demanding datasets may still benefit from a closer approximation.

In the previous chapter we showed that multitask learning may be more effective when the full set of parallel tasks is divided into smaller subsets (or clusters), and different sets of hyperparameters are used for different clusters. In future work we may incorporate the same 'task clustering' in the DHM presented in this chapter.

The current model features only continuous latent states, and all distributions are Gaussian. Interesting work may be done in the implementation of switching Kalman filters, discrete state variables and non-normal distributions.

Appendix 4.A The variational approach

In Section 4.3.1 we presented an approximation to our dynamic hierarchical model in the form of $Q(Z_{1..T}) = \prod_{i=0}^{n} Q_i(\boldsymbol{\theta}_{i,1..T})$. We defined the optimal approximation as the one that minimizes

$$\mathrm{KL}[Q,P] = \int dZ_{1..T} Q(Z_{1..T}) \left[\log Q(Z_{1..T}) - \log P(Z_{1..T}|Y_{1..T})\right] . \qquad (4.39)$$

For minimization with respect to (the parameters of) $Q_i(\boldsymbol{\theta}_{i,1..T})$ we need only be concerned with the part of the KL-divergence that depends on $Q_i(\boldsymbol{\theta}_{i,1..T})$. This part reads

$$\begin{aligned} \mathrm{KL}[Q_i, P] &= \int dZ_{1..T} Q(Z_{1..T}) \left[\log Q_i(\boldsymbol{\theta}_{i,1..T}) - \log P(Z_{1..T}|Y_{1..T})\right] \\ &\propto \int d\boldsymbol{\theta}_{i,1..T} Q_i(\boldsymbol{\theta}_{i,1..T}) \left[\log Q_i(\boldsymbol{\theta}_{i,1..T}) - \right. \\ &\qquad \left. \int dZ_{-i,1..T} Q_{-i}(Z_{-i,1..T}) \log P(Z_{1..T}|Y_{1..T})\right] , \end{aligned} \tag{4.40}$$

where $Q_{-i}(Z_{-i,1..T})$ is the product over all $Q_j(\boldsymbol{\theta}_{j,1..T})$ except for $j = i$. The last line is in fact the KL-divergence between $Q_i(\boldsymbol{\theta}_{i,1..T})$ and

$$\exp\left[\int dZ_{-i,1..T} Q_{-i}(Z_{-i,1..T}) \log P(Z_{1..T}|Y_{1..T})\right] . \tag{4.41}$$

Minimization of the KL-divergence with respect to $Q_i(\boldsymbol{\theta}_{i,1..T})$ therefore implies

$$Q_i(\boldsymbol{\theta}_{i,1..T}) = \exp\left\langle \log P(Z_{1..T}|Y_{1..T})\right\rangle_{Q_{-i}} . \tag{4.42}$$

Given this form for the approximating distributions, we presented an iterative optimization process, where in each step the KL-divergence between the approximating and the exact distribution was minimized with respect to one of the distributions $Q_i(\boldsymbol{\theta}_{i.1..T})$. We stated that the optimum for each $Q_i(\boldsymbol{\theta}_{i.1..T})$ can be interpreted as the posterior of a standard Markov model, provided that we make the proper variable transformations. In the following we show what transformations are required both for $Q_i(\boldsymbol{\theta}_{i,1..T})$, $i > 0$ and for $Q_0(\boldsymbol{\theta}_{0,1..T})$, the top-level DLM.

The variational approximation for the lower-level series with index i,

$$Q_i(\boldsymbol{\theta}_{i,1..T}) \propto \prod_t P(y_{i,t}|\boldsymbol{\theta}_{i,t}) \exp\left\langle \log P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t)\right\rangle_{Q_0} , \tag{4.43}$$

can be written as the posterior of a standard Markov model if we replace

$$\tilde{\boldsymbol{\theta}}_{i,t} = \boldsymbol{\theta}_{i,t} - \boldsymbol{\alpha}_t , \tag{4.44}$$

where

$$\boldsymbol{\alpha}_1 = \mathbf{0} \text{ (a vector of zeros)} , \tag{4.45}$$

$$\boldsymbol{\alpha}_t = A\boldsymbol{\alpha}_{t-1} + (\mathbb{1} - A)\left\langle \mathbf{M}_t\right\rangle \text{ for } t > 1 , \tag{4.46}$$

and

$$\tilde{y}_{i,t} = y_{i,t} - \mathbf{x}_{i,t}^T \boldsymbol{\alpha}_t . \tag{4.47}$$

Substitution in (4.43) yields

$$Q_i(\boldsymbol{\theta}_{i,1..T}) \propto P(\tilde{\boldsymbol{\theta}}_{i,1}) \prod_{t=2}^{T} P(\tilde{\boldsymbol{\theta}}_{i,t}|\tilde{\boldsymbol{\theta}}_{i,t-1}) \prod_{t=1}^{T} P(\tilde{y}_{i,t}|\mathbf{x}_{i,t}, \tilde{\boldsymbol{\theta}}_{i,t}) \,, \tag{4.48}$$

a time series model with states $\tilde{\boldsymbol{\theta}}_{i,t}$ and observations $\tilde{y}_{i,t}$.

We stated that the approximation for the top-level DLM $Q_0(\mathbf{M}_{1..T})$ can be interpreted as the posterior of a Markov model with evolution equation (4.3) and observation equation $P(\tilde{\mathbf{y}}_t|\mathbf{M}_t)$. This can be realized when we define observations

$$\tilde{\mathbf{y}}_t = n^{-1} \sum_i (\mathbb{1} - A)^{-1} (\langle \boldsymbol{\theta}_{i,t} \rangle - A \langle \boldsymbol{\theta}_{i,t-1} \rangle) \text{ for } t > 1 \tag{4.49}$$

$$\tilde{\mathbf{y}}_1 = \mathbf{0} \,, \tag{4.50}$$

covariate matrices $\tilde{C}_t$ that are $\mathbb{0}$ for $t = 1$ and $\mathbb{1}$ otherwise, and an observation covariance matrix defined through

$$\tilde{\Sigma}_y^{-1} = n(\mathbb{1} - A)^T \Sigma^{-1} (\mathbb{1} - A) \,. \tag{4.51}$$

Inserting these transformed parameters, we obtain

$$\begin{aligned} P(\tilde{\mathbf{y}}_t|\mathbf{M}_t) &= \prod_i \exp \langle \log P(\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_{i,t-1}, \mathbf{M}_t) \rangle_{Q_i} \\ &\propto \exp\left(-\frac{1}{2} \tilde{\mathbf{y}}_t^T \tilde{\Sigma}_y^{-1} \tilde{\mathbf{y}}_t + \tilde{\mathbf{y}}_t^T \tilde{\Sigma}_y^{-1} \tilde{C}_t \mathbf{M}_t - \frac{1}{2} \mathbf{M}_t^T \tilde{C}_t^T \tilde{\Sigma}_y^{-1} \tilde{C}_t \mathbf{M}_t \right) \\ &\propto P(\tilde{\mathbf{y}}_t|\boldsymbol{\theta}_{i,t}, \tilde{C}_t) \,, \end{aligned} \tag{4.52}$$

a normal observation equation with predictions

$$\tilde{\mathbf{y}}_t = \tilde{C}_t \mathbf{M}_t + \Theta_t, \;\; E(\Theta_t \Theta_t^T) = \tilde{\Sigma}_y \,. \tag{4.53}$$

Note that in the second line we dropped all terms that do not depend on $\mathbf{M}_t$, and added a term $\tilde{\mathbf{y}}_t^T \tilde{\Sigma}_y^{-1} \tilde{\mathbf{y}}_t$, which does not depend on $\mathbf{M}_t$ either.

Appendix 4.B Exact means in the variational approach

It can be proven that the means that are inferred from the variational approximation coincide with the exact means. In the variational approximation the KL-divergence between an approximating distribution and the exact distribution is minimized. We will show first that both distributions can be rewritten as a normal distribution over one large state that contains all latent states

present in the model. Second, we show that when the KL-divergence between these two normal distributions is minimized, their means coincide (but not their variances).

Both the exact and the approximating distribution can be written as one huge normal distribution of $\mathbf{Z}_{1..T}$, which is a vector of length $L = n_{\text{input}} \cdot (n+1) \cdot T$ and strings together all states at all times:

$$\mathbf{Z}_{1..T} = [\mathbf{Z}_1, \ldots, \mathbf{Z}_T] \,, \tag{4.54}$$

where $\mathbf{Z}_t$ is the super state for time t, defined in Section 4.2.1.

The exact posterior reads:

$$P(\mathbf{Z}_{1..T}|Y_{1..T}, \Lambda) = \mathcal{N}(\mathbf{m}, \Sigma_Z) \,, \tag{4.55}$$

where Σ_Z is a block matrix of dimension $L \times L$, defined through:

$$\Sigma_Z^{-1} = \begin{pmatrix} \Gamma_{0,0}^{-1} & \Gamma_{0,1}^{-1} & \ldots & \Gamma_{0,n}^{-1} \\ \Gamma_{1,0}^{-1} & \Gamma_{1,1}^{-1} & \ldots & \Gamma_{1,n}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_{n,0}^{-1} & \Gamma_{n,1}^{-1} & \ldots & \Gamma_{n,n}^{-1} \end{pmatrix} \,, \tag{4.56}$$

where the blocks $\Gamma_{i,j}^{-1}$ are $n_{\text{input}} \cdot T \times n_{\text{input}} \cdot T$ matrices corresponding to the inverse covariance of $\Theta_i = [\boldsymbol{\theta}_{i,1}, \ldots, \boldsymbol{\theta}_{i,T}]$ and $\Theta_j = [\boldsymbol{\theta}_{j,1}, \ldots, \boldsymbol{\theta}_{j,T}]$. The elements of this matrices follow from Equations 4.1 through 4.6 and Equations 4.10 through 4.14. The same equations define the means $\mathbf{m}$.

For the variational approximation we can define a similar distribution. The approximate posterior reads

$$\hat{P}(\mathbf{Z}_{1..T}|Y_{1..T}, \Lambda) = \mathcal{N}(\hat{\mathbf{m}}, \hat{\Sigma}_Z) \,, \tag{4.57}$$

with

$$\hat{\Sigma}_Z^{-1} = \begin{pmatrix} \hat{\Gamma}_{0,0}^{-1} & \mathbb{0} & \ldots & \mathbb{0} \\ \mathbb{0} & \hat{\Gamma}_{1,1}^{-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{0} \\ \mathbb{0} & \ldots & \mathbb{0} & \hat{\Gamma}_{n,n}^{-1} \end{pmatrix} \,, \tag{4.58}$$

where the blocks $\hat{\Gamma}_{i,i}^{-1}$ and the elements of $\hat{\mathbf{m}}$ are free to choose (provided that $\hat{\Gamma}_{i,i}$ is positive definite).

Note that for both the exact and the approximating distribution, it does not matter whether we write it in the form of a (series of) DLM(s), or as one large normal distribution. The inferred means, for example, are the same for both ways of writing.

The KL-divergence between Q and P has a mean-dependent term that reads

$$\mathrm{KL}_{\mathrm{mean}} = \frac{1}{2}\left(\mathbf{m} - \hat{\mathbf{m}}\right)^T \Sigma_Z \left(\mathbf{m} - \hat{\mathbf{m}}\right) , \tag{4.59}$$

and a term that depends only on the variances. Since there are no constraints on what values $\hat{\mathbf{m}}$ may take, it is clear that minimization of the KL-divergence, and therefore minimization of the above mean-dependent term with respect to $\hat{\mathbf{m}}$, infers that the approximating means $\hat{\mathbf{m}}$ are equal to the exact means $\mathbf{m}$.

The variance-dependent term reads

$$\mathrm{KL}_{\mathrm{variance}} = \frac{1}{2}\mathrm{Tr}\ \Sigma_Z^{-\frac{1}{2}} \hat{\Sigma}_Z \Sigma_Z^{-\frac{1}{2}} - \frac{1}{2}\log|\hat{\Sigma}_Z| . \tag{4.60}$$

Minimization of this term with respect to $\hat{\Sigma}_Z$ would yield exact marginal variances if the elements of $\hat{\Sigma}_Z$ were completely free to choose. This is however not the case: the variational approximation excludes dependencies between states $\boldsymbol{\theta}_{i,t}$ and $\boldsymbol{\theta}_{j.t'}$ for $i \neq j$, and the corresponding elements of $\hat{\Sigma}_Z$ must be zero. The approximate total variance $\hat{\Sigma}_Z$ can therefore not be equal to the exact variance Σ_Z. Further, the more involved form of the variance-dependent term of the KL-divergence keeps the marginal variance from coinciding with the exact marginal variance as well.

Appendix 4.C Exact means in the factorial approach

In Section 4.4 we stated that the means inferred through the factorial approximation are equal to the exact means as well. To show this, we first note that fixed points of expectation propagation as performed in Section 4.4 correspond to fixed points of the 'Bethe free energy' (see [42, 63]), which reads

$$\begin{aligned} F(\Psi, Q) \quad = \quad & -\sum_{t=1}^{T-1} \int d\mathbf{Z}_t Q_t(\mathbf{Z}_t) \log Q_t(\mathbf{Z}_t) \\ & + \sum_{t=1}^{T} \int dZ_{t-1,t} Q_{t-1:t}(Z_{t-1,t}) \log \left[\frac{Q_{t-1:t}(Z_{t-1,t})}{\Psi_t(Z_{t-1,t})} \right] , \end{aligned} \tag{4.61}$$

under constraints

$$\langle \mathbf{Z}_t \rangle_{Q_{t-1:t}} = \langle \mathbf{Z}_t \rangle_{Q_{t:t+1}} = \langle \mathbf{Z}_t \rangle_{Q_t} \tag{4.62}$$

and

$$\left\langle \mathbf{Z}_t^T \mathbf{Z}_t \right\rangle_{Q_{t-1:t}} = \left\langle \mathbf{Z}_t^T \mathbf{Z}_t \right\rangle_{Q_{t:t+1}} = \left\langle \mathbf{Z}_t^T \mathbf{Z}_t \right\rangle_{Q_t} , \tag{4.63}$$

where

$$Q_t(\mathbf{Z}_t) = \prod_{i=0}^{n} \lambda_{i,t}(\boldsymbol{\theta}_{i,t}) \mu_{i,t}(\boldsymbol{\theta}_{i,t}) \tag{4.64}$$

and

$$Q_{t-1:t}(Z_{t-1,t}) = \prod_{i=0}^{n} \lambda_{i,t-1}(\boldsymbol{\theta}_{i,t})\Psi_t(Z_{t-1,t})\mu_{i,t}(\boldsymbol{\theta}_{i,t}) \,, \tag{4.65}$$

as in Section 4.4, and Ψ_t is the potential defined in (4.12).

Next, we show that $F(\Psi, Q)$ can be written as a sum of three variants of a KL-divergence between an approximating distribution and the exact distribution. All three approximating distributions are Gaussian and feature the same means. We will then argue that minimization of a sum of such KL-divergences implies exact means for the factorial approximation.

Note first that we can express the KL-divergence between the exact posterior

$$P(Z_{1..T}|Y_{1..T}, \Lambda) \propto \prod_{t=1}^{T} \Psi_t(Z_{t-1,t}) \tag{4.66}$$

and the approximation

$$Q(Z_{1..T}) = \prod_{t} Q_t(\mathbf{Z}_t) \tag{4.67}$$

as

$$\begin{aligned}
\mathrm{KL}(Q(Z_{1..T})|P(Z_{1..T})) &= \int dZ_{1..T} \prod_{t'=1}^{T} Q_{t'}(\mathbf{Z}_{t'}) \log \frac{\prod_{t=1}^{T} Q_t(\mathbf{Z}_t)}{\prod_{t=1}^{T} \Psi_t(Z_{t-1,t})} \\
&= \sum_{t=1}^{T} \int d\mathbf{Z}_t Q_t(\mathbf{Z}_t) \log Q_t(\mathbf{Z}_t) - \\
&\quad \sum_{t=1}^{T} \int dZ_{t-1,t} Q_{t-1}(\mathbf{Z}_{t-1}) Q_t(\mathbf{Z}_t) \log \Psi_t(Z_{t-1,t}) \\
&= \sum_{t=1}^{T-1} \int d\mathbf{Z}_t Q_t(\mathbf{Z}_t) \log Q_t(\mathbf{Z}_t) + C - \\
&\quad \sum_{t=1}^{T} \int dZ_{t-1,t} Q_{t-1}(\mathbf{Z}_{t-1}) Q_t(\mathbf{Z}_t) \log \Psi_t(Z_{t-1,t}) \,,
\end{aligned} \tag{4.68}$$

where C is a constant, and can be ignored. (Each $\int d\mathbf{Z}_t Q_t(\mathbf{Z}_t) \log Q_t(\mathbf{Z}_t)$ is in fact a constant, but we keep the terms for $t = 1..T-1$ in for later use.)

Similarly, we can express the KL-divergence between the exact posterior and the approximation

$$Q_{\text{even}}(Z_{1..T}) = \prod_{t=\text{even}} Q_{t-1:t}(Z_{t-1,t}) \tag{4.69}$$

as

$$\begin{aligned}\mathrm{KL}(Q_{\mathrm{even}}(Z_{1..T})|P(Z_{1..T})) &= \int dZ_{1..T} \prod_{t'=\mathrm{even}} Q_{t'-1:t'}(Z_{t'-1,t'}) \log \frac{\prod_{t=\mathrm{even}} Q_{t-1:t}(Z_{t-1,t})}{\prod_{t=1}^{T} \Psi_t(Z_{t-1,t})} \\ &= \sum_{t=\mathrm{even}} \int dZ_{t-1,t} Q_{t-1:t}(Z_{t-1,t}) \log Q_{t-1:t}(Z_{t-1,t}) \\ &\quad - \sum_{t=\mathrm{even}} \int dZ_{t-1,t} Q_{t-1:t}(Z_{t-1,t}) \log \Psi_t(Z_{t-1,t}) \\ &\quad - \sum_{t=\mathrm{odd}} \int dZ_{1..T} \left(\prod_{t'=\mathrm{even}} Q_{t'-1:t'}(Z_{t'-1,t'}) \right) \log \Psi_t(Z_{t-1,t}) \,.\end{aligned} \tag{4.70}$$

The last line can be re-expressed as

$$-\sum_{t=\mathrm{odd}} \int dZ_{t-2,..,t+1} Q_{t-2:t-1}(Z_{t-2,t-1}) Q_{t:t+1}(Z_{t,t+1}) \log \Psi_t(Z_{t-1,t}) \,. \tag{4.71}$$

The expression $\log \Psi_t(Z_{t-1,t})$ contains terms proportional to Z_{t-1}, Z_t, $Z_{t-1}^T Z_{t-1}$, $Z_t^T Z_t$ and $Z_{t-1}^T Z_t$. The above expression therefore splits up in terms proportional to $\langle Z_{t-1} \rangle_{Q_{t-2:t-1}}$, $\langle Z_t \rangle_{Q_{t:t+1}}$, $\langle Z_{t-1}^T Z_{t-1} \rangle_{Q_{t-2:t-1}}$, $\langle Z_t^T Z_t \rangle_{Q_{t:t+1}}$ and $\langle Z_{t-1} \rangle_{Q_{t-2:t-1}}^T \langle Z_t \rangle_{Q_{t:t+1}}$. Due to the constraints of Equation 4.63 we can simplify the above expression to

$$-\sum_{t=\mathrm{odd}} \int dZ_{t-1,t} Q_{t-1}(\mathbf{Z}_{t-1}) Q_t(\mathbf{Z}_t) \log \Psi_t(Z_{t-1,t}) \,. \tag{4.72}$$

We can write down a similar expression for $\mathrm{KL}(Q_{\mathrm{odd}}(Z_{1..T})|P(Z_{1..T}))$. Carefully collecting all terms, we find that

$$\begin{aligned}F(\Psi, Q) &= -\mathrm{KL}(Q(Z_{1..T})|P(Z_{1..T})) \\ &\quad +\mathrm{KL}(Q_{\mathrm{even}}(Z_{1..T})|P(Z_{1..T})) + \mathrm{KL}(Q_{\mathrm{odd}}(Z_{1..T})|P(Z_{1..T})) \,,\end{aligned} \tag{4.73}$$

a linear combination of the KL-divergences of three approximating distributions (the fully factorized distribution and two factorizations into two-slice marginals) to the exact posterior. By design, the means of the factorized distribution are equal to the means of the two two-slice distributions at convergence of the algorithm described in Section 4.4. Following Appendix 4.B we can write out the mean-dependent terms of each of the three KL-divergences (which all three feature the same exact mean $\mathbf{m}$, and have by definition the same approximate mean $\hat{\mathbf{m}}$), and obtain an expression of the form of Equation 4.59. Minimization of this term infers that the means in the factorial approximation are exact as well.

Chapter 5

Clustering ensembles of neural network models

Abstract. We show that large ensembles of (neural network) models, obtained e.g. in bootstrapping or sampling from (Bayesian) probability distributions, can be effectively summarized by a relatively small number of representative models. In some cases this summary may even yield better function estimates. We present a method to find representative models through clustering based on the models' outputs on a data set. We apply the method on an ensemble of neural network models obtained from bootstrapping on the Boston housing data, and use the results to discuss bootstrapping in terms of bias and variance. A parallel application is the prediction of newspaper sales, where we learn a series of parallel tasks. The results indicate that it is not necessary to store all samples in the ensembles: a small number of representative models generally matches, or even surpasses, the performance of the full ensemble. The thus obtained clustered representation of the ensemble is much better suitable for qualitative analysis, and will be shown to yield new insights into the data.

Adapted from: B. Bakker and T. Heskes. Clustering ensembles of neural network models, *Neural Networks*, 16(2):261–269.

5.1 Introduction

In neural network analysis we often find ourselves confronted with a large ensemble of models trained on one database. One example of this is resampling, which is a popular approach to try and obtain better generalization performance with nonlinear models. Individual models are trained on slightly different samples of the available data set, which are generated e.g. by bootstrapping or cross-validation (see e.g. [30]). Another example can be found in the Bayesian approach, which creates a probability distribution over all possible models, which may be sampled from by Markov Chain Monte Carlo procedures [65].

In both cases a considerable number of network representations, or models, may be needed to catch the fine nuances of the system. For complex problems, the number of models may even be too high to keep a good overview of the ensemble and special transformations will be required to summarize it, preferably without loss of information. The clustering procedure that we will present will help to understand such problems better, and may be a valuable tool in their analysis.

Clustering can be seen as the representation of a large collection W by a smaller collection of representative entities, M. The components of W and M may be anything: locations on a map, people, words, models, etc. The type of the elements may even vary from set W to set M, or within sets (see e.g. [20]). The only requirement is that there exists a distance function $D(W, M)$ indicating how much sets W and M differ from each other, or rather how much information is lost in the conversion from W to M.

Since W and M may contain any kind of elements, the method of clustering may well be used to find a workable representation of any oversized ensemble of (neural network) models. Taking the elements of W to be the models in the original (large) ensemble, represented in the case of neural networks by their weights, biases and overall structure (number of hidden layers, transfer functions, etc.), M can be a smaller set of networks best representing the features contained in W.

In Section 5.2 we will review the clustering method outlined by Rose *et al* [73]. Their method, based on the principles of deterministic annealing as first described in [48], was shown to yield good results for the clustering of two-dimensional data with a Euclidean distance function. We will generalize this method for use with other data types and distance functions, and use it to cluster models (e.g. neural networks). In Section 5.3 we describe the algorithms that we use to implement the method.

Although clustering in weight space has been used in network analysis [76], model clustering as described in this thesis is, to the best of our knowledge, a new approach. In our approach the distance function $D(W, M)$ is based on model outputs instead of model parameters. We feel this approach is more

intuitive, since model outputs on a known database provide a more direct representation of the models' characteristics than the more abstract model parameters themselves.

The expression of the distance in terms of model outputs, combined with a suitable distance formula (e.g. sum-squared distance), further allows us to lower the computation time required to perform clustering considerably. We accomplish this by expressing not only the distance, but also the representative models themselves in terms of their outputs. We will show in Section 5.3 how this leads to an algorithm that is much faster than one finding representative models in terms of their model parameters. The final result, however, needs to be in the form of representative *models*, and not *model outputs*. In Section 5.4 we discuss several methods to make the translation from model outputs to model parameters. We compare the methods in terms of computation time and quality of the obtained representatives.

We apply our method on two databases in Section 5.6.2. The first, containing the well-known Boston housing data, serves as a benchmark problem to study the effect of bootstrapping. We apply the clustering algorithm that is described in this chapter on ensembles of models obtained through bootstrapping, and use the resulting clusters to illustrate the effect of bootstrapping in terms of bias/variance reduction.

The other database concerns the prediction of single-copy newspaper sales in the Netherlands. This can be represented as a series of parallel tasks (outlets at different locations), which we optimize through multitask learning. In multitask learning (see e.g. [40]) one is presented with a (preferably large) number of parallel tasks, e.g. predicting student test results for students in a series of schools, or survival analysis on patients in a series of hospitals. Such a series of tasks can be represented by a series of models, each trained on one of the tasks. Although any one of these models may be overfitting its training data to some extent, the ensemble of all models may yield a good estimation of the underlying function. We show that a clustered representation of this ensemble can give valuable new insights into the nature of the problem. Also, the knowledge about the grouping of the models into tasks may allow the clustered solution to reach a significantly lower prediction error than the full ensemble.

5.2 Clustering by deterministic annealing

5.2.1 Notation

Suppose we have N_W models, fully characterized through their parameters $\mathbf{w}_i$, $i = 1 \ldots N_W$. We define N_M other models, which we will refer to as cluster centers, denoted $\mathbf{m}_\alpha$, $\alpha = 1 \ldots N_M$. $D(\mathbf{w}_i, \mathbf{m}_\alpha)$ is the distance of model i to cluster center α. The distance need not be symmetric, i.e. $D(\mathbf{w}, \mathbf{m})$ may

be different from $D(\mathbf{m}, \mathbf{w})$[1]. We do however require that $D(\mathbf{m}, \mathbf{w}) \geq 0$ and $D(\mathbf{m}, \mathbf{m}) = 0$. The models considered in this chapter are feedforward models that are optimized through supervised training.

We assume distances of the form

$$D(\mathbf{w}_i, \mathbf{m}_\alpha) = \sum_\mu d(y(\mathbf{w}_i, \mathbf{x}^\mu), y(\mathbf{m}_\alpha, \mathbf{x}^\mu)) \,, \tag{5.1}$$

for some distance measure $d(y_1, y_2)$, where $y(\mathbf{w}_i, \mathbf{x}^\mu)$ is the output of a model with parameters $\mathbf{w}_i$ on an input $\mathbf{x}^\mu$. Each model is supposed to have the same input. Since, once the inputs $\mathbf{x}^\mu$ are given, the clustering procedure depends on the models $\mathbf{w}_i$ only through the outputs $y(\mathbf{w}_i, \mathbf{x}^\mu)$, we can compute these outputs in advance.

5.2.2 The derivation of 'free energy'

We define variables $p_{i\alpha}$ as the probability that model i belongs to cluster α. The distances between models $\mathbf{w}_i$ and cluster centers $\mathbf{m}_\alpha$ are given by $D(\mathbf{w}_i, \mathbf{m}_\alpha)$. We assume that the models $\mathbf{w}_i$ are given, but that the probabilities $p_{i\alpha}$ and cluster centers $\mathbf{m}_\alpha$ are still free to choose. One of the goals of clustering is to put the cluster centers such as to minimize the average distance of the models to the cluster centers, i.e., to find a low average energy

$$E(\{m\}, \{p\}) = \sum_{i\alpha} p_{i\alpha} D(\mathbf{w}_i, \mathbf{m}_\alpha) \,, \tag{5.2}$$

where $\{m\}$ and $\{p\}$ represent the full sets $\mathbf{m}_\alpha$ and $p_{i\alpha}$ $\forall_{i\alpha}$. In this framework the average energy is the average over the distances between models and cluster centers, weighted by $p_{i\alpha}$. For fixed cluster centers $\mathbf{m}_\alpha$, minimizing the average energy would correspond to assigning each model to its nearest cluster center with probability one. A proper way to regularize this is through a penalty term of the form

$$S(\{p\}) = -\sum_{i\alpha} p_{i\alpha} \log p_{i\alpha} \,, \tag{5.3}$$

the discrete version of the Shannon entropy, which is the only quantity which is positive, increases with increasing uncertainty, and is additive for independent sources of uncertainty [48]. Maximizing $S(\{p\})$ therefore favors a state without any structure, i.e. $p_{i\alpha} = p_{i'\alpha'}$ $\forall_{i,i',\alpha,\alpha'}$, which corresponds to the notion that we have no prior knowledge about the structure of the clusters.

[1]Note that $D(\mathbf{w}, \mathbf{m})$ is not a distance function in the mathematical sense, but rather a measure of the difference between $\mathbf{w}$ and $\mathbf{m}$. However, since the concept of clustering is intuitively best understood in terms of positions and distances, we will still refer to $D(\mathbf{w}, \mathbf{m})$ as a distance.

We introduce a regularization parameter T, weighting the two different terms, to arrive at the 'free energy'

$$F(\{m\},\{p\}) = E(\{m\},\{p\}) - TS(\{p\}) \,. \tag{5.4}$$

Minimizing $F(\{m\},\{p\})$ can be seen as a search for the best compromise between a low average distance (minimizing $E(\{m\},\{p\})$) and not imposing *too* much structure on the system (maximizing $S(\{p\})$). For any choice of cluster centers $\mathbf{m}_\alpha$, the probabilities $p_{i\alpha}$ minimizing the free energy $F(\{m\},\{p\})$ read (under the constraint $\sum_\alpha p_{i\alpha} = 1 \ \forall_i$)

$$p_{i\alpha}(\{m\}) = \frac{\mathrm{e}^{-\beta D(\mathbf{w}_i,\mathbf{m}_\alpha)}}{\sum_{\alpha'} \mathrm{e}^{-\beta D(\mathbf{w}_i,\mathbf{m}_{\alpha'})}} \tag{5.5}$$

with $\beta = 1/T$. Substitution of this result into the free energy then yields

$$F(\{m\}) = F(\{m\},\{p(\{m\})\}) = -\beta^{-1} \sum_i \log \sum_\alpha \mathrm{e}^{-\beta D(\mathbf{w}_i,m_\alpha)} \,. \tag{5.6}$$

Equation 5.6 is equivalent to the result presented in [73]. The main difference between our derivation and the derivation made by Rose *et al* is the role of the parameter β. Here, as in [18], it is just a regularization parameter, that can be chosen in advance. In [73] an average energy $\langle E \rangle$ is defined, such that (like in statistical physics theory) β is a Lagrange multiplier that must be tuned to ensure that $\langle E \rangle$ stays constant.

5.3 Annealing and the EM algorithm

The annealing process finds cluster centers $\mathbf{m}_\alpha$ minimizing the free energy $F(\{m\})$ for increasing values of β. We start with β close to zero and a large number of cluster centers $\mathbf{m}_\alpha$[2]. Such a low value of β will strongly favor the entropy part of the free energy, resulting in a solution where all $\mathbf{m}_\alpha$ are identical (so $p_{i\alpha} = p_{i'\alpha'} \ \forall_{i,i',\alpha,\alpha'}$). When the new $\mathbf{m}_\alpha$ have been found, we increase β and again minimize the free energy. These steps are repeated until the balance between average energy and entropy has shifted enough to warrant multiple clusters; at this point a *phase transition* occurs, and the cluster centers are divided over two separate solutions. More and more clusters will appear when β is increased further, until we have reached a satisfactory number of clusters and we terminate the process.

[2]The number of cluster centers is infinite in theory; in practice we implement a sufficiently large number of clusters at each point in the process. The exact choice for this number will be elaborated on in Section 5.4.

We find cluster centers $\mathbf{m}_\alpha$ (for each value of β) through minimization of the free energy $F(\{m\})$. This corresponds to solving a series of coupled equations:

$$\frac{\partial F(\{m\})}{\partial \mathbf{m}_\alpha} = \sum_i p_{i\alpha} \frac{\partial D_{i\alpha}}{\partial \mathbf{m}_\alpha} = 0 \quad \forall_\alpha \,, \tag{5.7}$$

where the equations for different $\mathbf{m}_\alpha$ are interdependent through the normalization of $p_{i\alpha}$, which is a function of all $\mathbf{m}_\alpha$. We solve this system of equations using an expectation-maximization (EM) algorithm, a full description of which can be found in [75].

In the expectation step of the EM algorithm the probabilistic assignments $p_{i\alpha}$, as given by Equation 5.5, are calculated. In the maximization step we find new cluster centers $\mathbf{m}'_\alpha$ such that:

$$\mathbf{m}'_\alpha = \underset{\mathbf{m}_\alpha}{\operatorname{argmax}} \left[-\sum_i p_{i\alpha} D(\mathbf{w}_i, \mathbf{m}_\alpha) \right] \quad \forall_\alpha \,. \tag{5.8}$$

A solution for $\mathbf{m}'_\alpha$ can be obtained from any gradient descent algorithm on $\sum_i p_{i\alpha} D(\mathbf{w}_i, \mathbf{m}_\alpha)$ starting from the current $\mathbf{m}_\alpha$.

Our aim is to summarize an ensemble of neural networks through a smaller set of networks with a similar architecture, and Equation 5.8 is expressed in terms of the parameters of these networks. However, the above description also applies to 'model free' clustering, solely based on the outputs on examples $\mathbf{x}^\mu$. The maximization step for model free clustering (where we now maximize with respect to $\mathbf{y}_\alpha = [y(\mathbf{m}_\alpha, \mathbf{x}^1), y(\mathbf{m}_\alpha, \mathbf{x}^2), \ldots, y(\mathbf{m}_\alpha, \mathbf{x}^N)]$) can be very simple: for suitable distance functions (e.g. the sum-squared error or the cross-entropy error[3]) it is simply

$$\mathbf{y}'_\alpha = \sum_i p_{i\alpha} \mathbf{y}_i \,. \tag{5.9}$$

Model free clustering often makes the maximization step (Equation 5.8) less complex, and the clustering algorithm much faster.

Model based clustering can still benefit from this simplified maximization step. We simply ignore the model parameters $\mathbf{w}_i$ during (parts of) the clustering process, and use the corresponding model outputs $\mathbf{y}_i$ for model free clustering. Obviously, the 'cluster outputs' need to be translated back to model parameters in the end (through optimization on the cluster center outputs $\mathbf{y}_\alpha$ and the inputs $\mathbf{x}^\mu$ that were used for clustering), but we may still gain a tremendous speed-up of the annealing process. In the next section we discuss several alternative procedures to combine model free clustering and translation back to model parameters.

[3] $d(y(\mathbf{w}_i, \mathbf{x}^\mu), y(\mathbf{m}_\alpha, \mathbf{x}^\mu)) = y(\mathbf{w}_i, \mathbf{x}^\mu) \log \frac{y(\mathbf{w}_i, \mathbf{x}^\mu)}{y(\mathbf{m}_\alpha, \mathbf{x}^\mu)} + [1 - y(\mathbf{w}_i, \mathbf{x}^\mu)] \log \frac{1 - y(\mathbf{w}_i, \mathbf{x}^\mu)}{1 - y(\mathbf{m}_\alpha, \mathbf{x}^\mu)}$

5.4 Computational issues

Deterministic annealing. The annealing process starts at $\beta = 10$ (for the data examined in this chapter). We first allow two clusters, which are initiated by adding random noise to the average of the model outputs over a full ensemble of models. After the EM algorithm described in the previous section has converged, we compare the distance between the two cluster centers to the distance between the models in the corresponding clusters. If the former is relatively small, the two cluster centers merge (we dispose of one of the two, approximately identical, clusters), otherwise a new cluster is accepted (we keep both clusters). For the initial, low value of β, the two clusters always merge. The annealing parameter β is increased by 1 after each merging or acceptation of a new cluster. After the first phase transition (when the algorithm has accepted two separate clusters), we split each of the corresponding representative model outputs in two by adding random noise. Thus we allow the next transition into three or four clusters to occur. Generally, only one new cluster is gained in each transition. When the algorithm does 'skip' intermediate numbers of clusters (e.g. goes from 3 clusters to 5 clusters after one increment of β), the increase in β (Δ_β) is temporarily lowered until all intermediate cluster numbers are found, or Δ_β drops below 0.01. This steady growth from one to a maximum number of clusters can be compared to 'greedy mixture learning' [84].

When to retrain. A downside of model free clustering of elements that are in fact models is that cluster centers are found directly in the space of model outputs. Such cluster centers generally do not have a generating model of the same architecture as the models in the ensemble. We therefore need a retraining step to translate the cluster centers back to actual models. To retrain a 'cluster model', we split the model's outputs, which were found in the clustering process, and the inputs that were used to generate these outputs into a training set and a validation set. Retraining then proceeds in exactly the same way that the original networks were trained: we minimize the mean squared error on the training set, and stop training when the error on the validation set starts to increase. We can make several choices for the point in the process where we make this translation:

1. Retrain after clustering. The simplest option is to perform the entire clustering process solely in terms of model outputs, and train back the actual representative models at the end, when all cluster centers are found. In doing this however, we disrupt the clustering we found: the models obtained through retraining may not be the optimal representatives, even though the cluster centers in terms of outputs were. Therefore, after retraining we could implement a short reclustering sequence, where now in each EM-step we retrain the networks to fit the cluster center outputs. Model free clustering then serves as an 'initialization' for model based clustering.

2. Cluster in terms of parameters. The other extreme is to perform clustering in model space entirely, varying only the parameters of the representative models. Although this makes each step considerably more time consuming, it does prevent the cluster centers from wandering into areas of output space that cannot be reached by the actual models. Cluster centers obtained from the previous approach that do stray into this area, may be very difficult to approximate by the applied models. In this case the final approximations may not only take a lot of extra time, but (e.g. due to local minima near the initialization for retraining the model) may also yield qualitatively inferior results.
3. Retrain during clustering. A reasonable compromise might be to retrain the model after every N EM-steps, where N is large enough not to unacceptably slow down the process, but small enough to keep the cluster centers in the right area. N can be kept constant throughout the clustering process, or vary, when for example we retrain for each new value of β, or each time a new cluster is accepted. In Section 5.6.3 we compare several approaches in terms of computation time and accuracy.

5.5 Bootstrapping and multitask learning

In bootstrapping, instead of training one model on the complete (training) data, we resample the data set multiple times, and train one model for each resampling. Resampling is done by drawing N samples from a training set of N instances, where each instance may be drawn more than once. In this way, the ensemble of models reflects the variability of the data under review. Breiman [13] describes how bootstrap ensembles can be used for prediction through a procedure called *bagging* (*b*ootstrap *agg*regat*ing*). If the data contains only global similarities, the models in the ensemble will be centered around one average model, and bootstrapping only serves to obtain a more unbiased version of this model. If however the data contains multiple local similarities, bootstrapping provides a way to include corresponding local summaries in the final predictions, which is impossible for any one model. In both cases, instead of using the entire ensemble, we may need only the local summaries, and a way to weigh them. This is provided by the clustering algorithm described in the previous sections. The cluster based prediction on a new input $\mathbf{x}^\nu$ reads:

$$\tilde{y}^\nu = \sum_\alpha p_\alpha y(\mathbf{m}_\alpha, \mathbf{x}^\nu), \text{ with } p_\alpha = \frac{1}{N}\sum_{i=1}^{N} p_{i\alpha}\,, \tag{5.10}$$

a weighted sum over the representative model outputs, where the weights are determined by the 'effective' number of models in each cluster.

Multitask learning, or 'learning to learn', makes use of the fact that a series of similar tasks may share a common underlying structure. Instead of learning each of these tasks separately, the tasks are forced to share their knowledge, and learn from each other. One way to implement this is a 'hard sharing' of part of the model parameters (see e.g. [23]). In this chapter we allow knowledge sharing through clustering (see [20] for a similar approach). For each of the tasks we train one or more models, which will generally be strongly biased to the part of the training set corresponding to that task. Task clustering can then be applied to represent strongly similar tasks by a common cluster. A predicted new output for task t, $\tilde{y}_t^\nu$, changes slightly to include knowledge of the partition of the data into tasks:

$$\tilde{y}_t^\nu = \sum_\alpha p_{t\alpha} y(\mathbf{m}_\alpha, \mathbf{x}_t^\nu), \text{ with } p_{t\alpha} = \frac{1}{n_t} \sum_{i \in t} p_{i\alpha} \,, \tag{5.11}$$

where n_t is the number of models trained for task t, and where the weights $p_{t\alpha}$ for task t depend only on the corresponding models trained on this task.

Note that we apply the same model clustering procedure to two very different problems. The models obtained in bootstrapping are interchangeable, and cannot *a priori* be assigned to different clusters. The models for multitask learning are each trained on a specific task, and can therefore be 'labeled'. This difference is apparent in Equations 5.10 and 5.11, where in the former all models contribute, whereas in the latter a model's contribution depends on its label.

5.6 Results

5.6.1 Description of the data

Boston housing. The Boston housing problem concerns the prediction of housing values in the suburbs of Boston, based on 13 inputs including e.g. per capita crime rate and nitric oxides concentration. The database contains 506 examples. For more information, see [34]. We preprocessed the data by scaling each variable (both input and output) to have zero mean and unit variance.

Prediction of newspaper sales. We also apply our method on a database of single-copy sales figures for one of the major Dutch newspapers. The database contains the numbers of newspapers sold on 156 consecutive Saturdays, at 343 outlets in The Netherlands. Inputs include recent sales (-4 to -6 weeks), last year's sales (-51 to -53 weeks), weather information (temperature, wind, sunshine, precipitation quantity and duration) and season (cosine and sine of scaled week number). The responses are the realized sales figures. All covariates and responses are scaled per outlet to zero mean and unit standard deviation.

5.6.2 Clustering and prediction

Methods. For both databases we trained ensembles of neural networks. These networks had one hidden layer and biases on the hidden and output units. The hidden units had hyperbolic tangents as transfer functions; the output units were linear. For the Boston housing data we trained ensembles of models with numbers of hidden units varying from 1 to 14 (one type of model in each ensemble). For the newspaper data we only trained networks with two hidden units, since we know from past experience that these yield the best results for this particular data set.

The clustering algorithm was applied to 10 independent ensembles of 50 models, obtained by bootstrapping. To create an ensemble of models we made a random 2:1 split of the database, where the larger part was used for training (training set) and the smaller for testing the final result (test set). Each of the 50 models was optimized on a random resampling of the training set: for a set of n_{train} samples we took n_{train} random draws where each data item (an input with its corresponding output) could be drawn more than once. This resampled set was used for the actual optimization (for which we applied a conjugate gradient method), and the undrawn part of the training set was used to implement early stopping. Larger ensembles did not improve performance, smaller ensembles yielded inferior results.

The clustering algorithm was performed for each ensemble. We predicted the outputs in the smaller part of the database (the test set) through Equations 5.10 and 5.11. The predictions on the Boston housing data were compared to prediction through bagging (taking the average over the outputs of all models in the ensemble on a new input). For the newspaper data, for each split of the data we also trained one network similar to those described previously, with two hidden units, but with one output for each task (outlet). This means that all tasks shared the same input-to-hidden weights of the network, but had independent hidden-to-output weights. See e.g. [23] and Chapter 3 of this thesis for similar work. All prediction results are rated through their sum-squared error on the test set.

Boston housing. We modeled the Boston housing problem by bootstrapping with multilayered perceptrons with their numbers of hidden units varying from 1 to 14. The models within one bootstrap ensemble always had the same numbers of hidden units. Each ensemble was clustered up to the point where no more improvement in sum-squared error was gained by adding more cluster centers. Figure 5.1 shows the sum-squared error for the clustered ensemble as a function of the number of cluster centers, for the full ensemble (through bagging) and the average sum-squared error of single models in the ensemble, for models with 2, 8, 12 and 14 hidden units. It can be seen that for more complicated models (more hidden units) less cluster centers are needed to

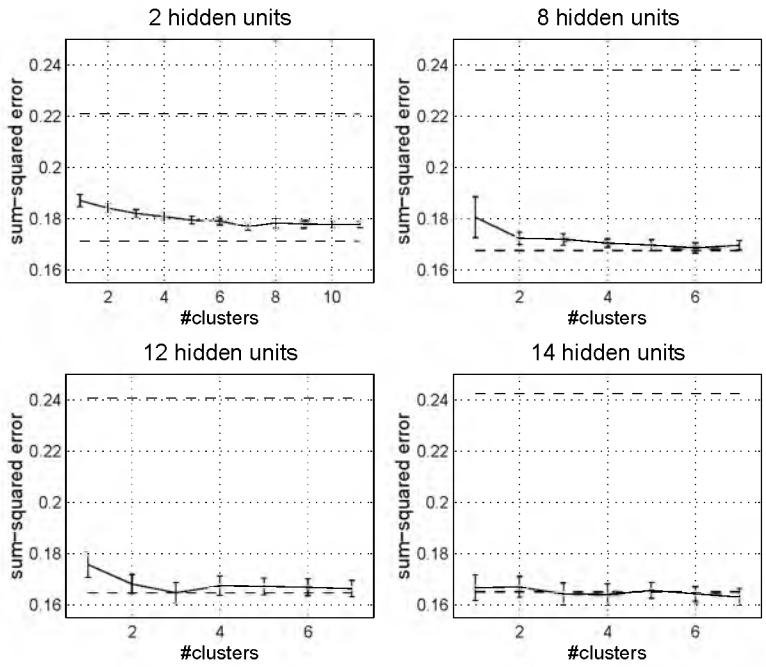


Figure 5.1: Boston housing data: prediction error as a function of the number of clusters for models with 2 (upper left panel), 8 (upper right panel), 12 (lower left panel) and 14 hidden units (lower right panel). The lower horizontal lines in each plot indicate the prediction error corresponding to the full bootstrap ensemble. The upper horizontal lines indicate the average error made by the single models in the ensemble. The error bars are calculated based on the difference between the error of the reduced ensembles (cluster centers) and the full ensemble.

match the performance of the full ensemble. This progression from many cluster centers to one can be understood in terms of bias and variance.

In general, we can say that errors due to (large) variance occur when a model can in fact adequately represent the (hypothetical) data generating function, but still optimizes to the wrong model because of the limited amount of training data. Errors due to bias, on the other hand, occur when the model is not sufficiently sophisticated to match the data generating function, and must settle for the closest approximation. For one simple model it is impossible to give an adequate representation of the hidden (complex) function underlying the data. Bootstrapping in this case serves to find an ensemble of models that, when put together, can closely approximate this function. Since in this case the bootstrapping ensemble contains multiple significantly different models,

multiple cluster centers are required to represent the ensemble. Bootstrapping then serves to reduce the bias in the model, since the summation over multiple networks in fact yields a more complex model.

One sufficiently complex model however, may be sufficient to represent the unknown function that generated the data. In this case, bootstrapping serves to reduce the overfitting that may occur when one such model is trained on the full data. Although for complicated models many bootstrap samples may be needed to obtain a low variance estimation, in the end the full ensemble can be represented by just one cluster center. More discussion on the effectiveness of bootstrapping and ensemble learning can be found in [15, 29].

One MLP with 14 hidden units appears to be sufficiently complex to describe the Boston housing problem: the one cluster representation performs as well as the full ensemble. Clustering of the two hidden units ensemble appears not to be able to match the performance of the full ensemble. We expect this failure to be due to the fact that in this case, where a large number of representative models is needed, the annealing process needs to reach relatively high values of β. In this regime the algorithm gets too 'greedy', and rather than detecting new, relevant clusters, it creates cluster centers that coincide with models in the ensemble, as can be seen in the lower part of the upper left panel in Figure 5.2.

Prediction of newspaper sales. Prediction error for the newspaper data (Figure 5.3) decreases strongly until the 3-cluster solution, and then slowly converges to a minimum, which is significantly lower than the error made by the multitask learning network (one output for each outlet, and shared input-to-hidden weights). Averaging over all models in the full ensemble (as in bootstrapping) yields rather poor results in this case. The clustering obtained on the newspaper data does not only improve prediction, but also reveals an interesting structure, which was hidden in the data. Figure 5.4 shows the outlets in Holland that are assigned to clusters 1, 2 and 3 with probability $p_{i\alpha} > 0.9$. It can be seen that the outlets in the first cluster tend to be near the beach, and in the eastern part of Holland, touristic spots without many large cities. The outlets in cluster 2 center around the 'Randstad' (Amsterdam and other relatively large Dutch cities). The outlets in cluster 3 are spread all around, and can be considered 'undetermined'. Figure 5.5 (a Hinton diagram [10]) plots the 'sensitivity' (the derivative of the model output with respect to a model input, averaged over a set of training samples) of the representative models for each cluster. The figure clearly shows that the first of these models (corresponding to the 'touristic' cluster) is especially sensitive to the inputs corresponding to last year's sales and season, whereas for the second model ('city cluster') short term sales are weighed more heavily. The third, 'undetermined' model features much less pronounced sensitivities. This clustering, which makes intuitive sense, was obtained without any information

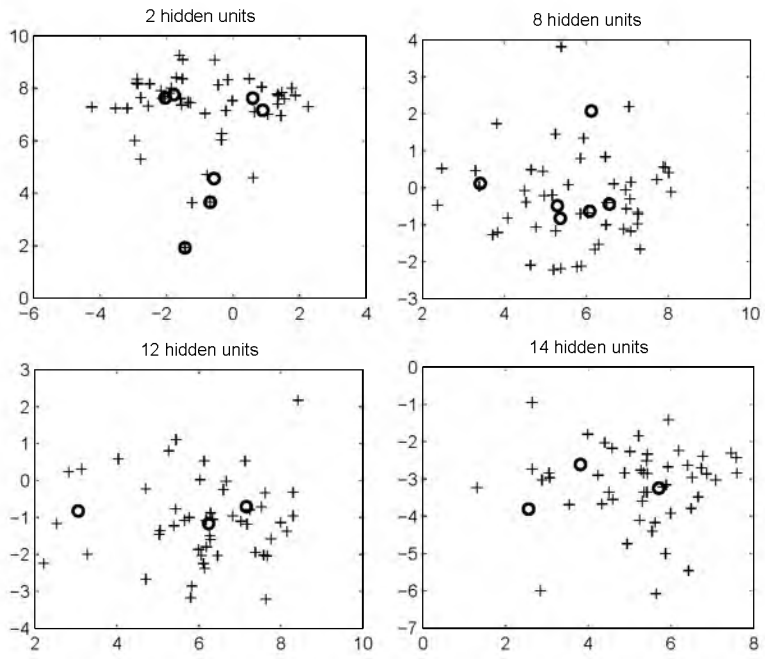


Figure 5.2: Clustering of the Boston housing data. Each panel plots the principal components of the outputs of both bootstrap models (crosses) and representative models (circles). For the 2 hidden units ensemble we show the 7 cluster solution, 6 clusters for the 8 hidden units ensemble and 3 for the remaining ensembles. Note that in the 2 hidden unit ensemble representative models and bootstrap models start to coincide.

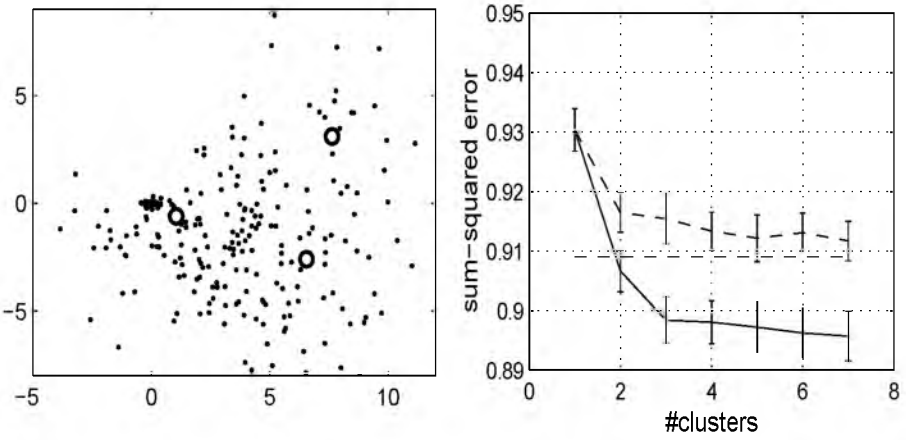


Figure 5.3: Clustering of the newspaper data. The left panel plots the principal components of the outputs of both bootstrap (dots) and representative models (circles). The right panel shows the prediction error as a function of the number of clusters. The horizontal line represents the prediction error that is made by the multitask learning network (one output for each outlet), the other dashed line shows the K-means clustering results, the solid line shows the results from the clustering method described in this chapter. The error bars are calculated based on the difference between the prediction error corresponding to averaging over the cluster centers and those made by the larger network.

with respect to city size or level of tourism, and is consistent with the earlier results of Chapter 3.

Other methods. Model clustering can in many cases be implemented through less involved clustering algorithms, such as K-means or nearest-neighbor clustering. We repeated the model clustering process and predictions described in the previous paragraphs, where this time we applied (randomly initialized) K-means clustering to the model outputs, instead of deterministic annealing. The performance of the simpler K-means clustering method on the Boston housing database was not significantly worse than that of the more involved method. Note however that in this case only 50 data vectors were clustered; more complex databases may still benefit from the more involved algorithm described in Section 5.3. See [62] for examples.

The newspaper example, however, does benefit from the more involved algorithm. The prediction error, shown in Figure 5.3, is significantly higher after K-means clustering. This effect is partly due to the fact that determinis-

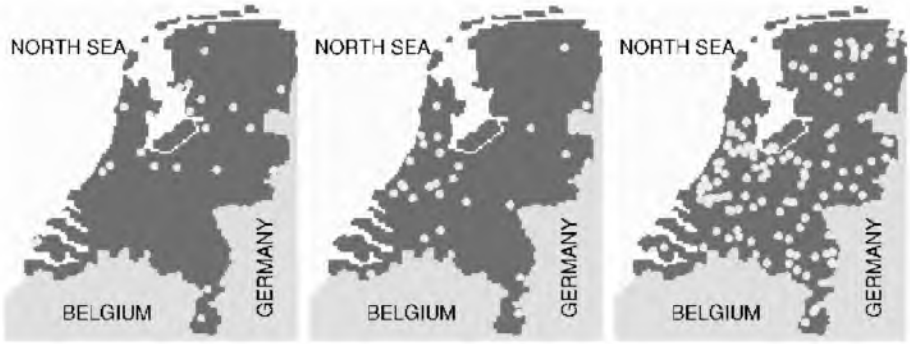


Figure 5.4: Geographical clustering of the newspaper outlets. Circles mark outlets assigned with weight larger than 0.9 to either the 'seasonal' cluster (left panel), the 'short term' cluster (middle panel) or the 'undetermined' cluster (right panel).

tic annealing yields a 'soft' clustering, where in the case of multitask learning for each task we get a distribution over the full set of clusters, instead of a hard assignment to one cluster; when for the deterministic annealing method we assign each task to its 'most likely' cluster (i.e. task t is assigned to the cluster α corresponding to the highest $p_{t\alpha}$), its performance becomes significantly worse (but is still better than the performance of K-means). The other part of the improvement most probably is due to the larger number of samples, which makes a more complex (and specificly a greedy) clustering algorithm more effective.

We also looked at a selection method, as an alternative to clustering. This alternative would use a (small) selection of models from the existing ensemble instead of cluster centers. We made a random 2:1 split of the original training data into a training set and a validation set, and trained an ensemble on the training set. We selected models from this ensemble through a method much like the 'Tabu method' (see [72]). Here, we start out with a random selection of n models from the ensemble (n being the number of models we eventually want to have in our selection; in our case $1 \leq n \leq 7$). In each step we consider all possible subsets created through removing one model from the selection and adding another (out of the ensemble) to it, and accept the subset with the lowest sum-squared error on the validation set. This process is continued for a fixed number of steps, and continues even when the validation error rises from one set to the other. Eventually, we accept that selection that over the course of the algorithm showed the lowest validation error. Although this method may often be less time consuming than the clustering method, the results in terms of prediction error were significantly worse than those of

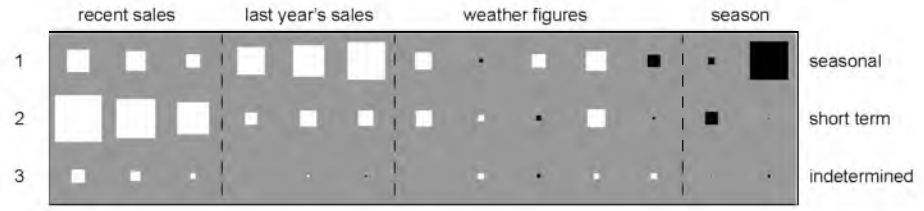


Figure 5.5: Sensitivity of the representative models to the model inputs. White squares represent positive dependencies, black squares negative dependencies. Larger squares represent stronger effects.

the method described in this chapter in each instance except for the Boston housing example with two hidden units, where they were equivalent.

5.6.3 The influence of frequent retraining

The results from Section 5.6.2 were obtained by retraining the models at the end of the clustering procedure. We mentioned in Section 5.4 that the moment and frequency of retraining the representative models may have its influence on the final solutions. Therefore we repeated the above simulations on newspaper sales for clustering with more frequent retraining, where we tried retraining either for each new β or after addition of each new cluster. Table 5.1 shows the simulation time and prediction error for each of these approaches. The values are averages over 10 independent runs, as in Section 5.6. Simulations on the Boston housing data yielded similar results. It can be seen that, at least for the databases used in this chapter, retraining at the end of the clustering procedure has no adverse effect on the quality of the methods predictions, and takes significantly less computation time. Still, other databases might produce other results.

As a benchmark we also applied the original EM algorithm outlined in Section 5.3, i.e. where the maximization step itself involves fitting the model parameters of the cluster centers. Even here, where we do not lose any accuracy due to retraining, the results are still not significantly better, whereas the computation time is considerably longer than for any of the retraining methods.

5.7 Discussion

In this chapter we have presented a method to summarize large ensembles of models to a small number of representative models. We have shown that

method	sum-squared error	required time (min)
retrain afterwards	0.897 ± 0.011	9.0 ± 1.6
retrain per cluster	0.897 ± 0.005	11.7 ± 2.2
retrain per β	0.897 ± 0.005	28 ± 4
cluster on parameters	0.897 ± 0.005	116 ± 19

Table 5.1: Average sum-squared error on the test set and required computation time for different moments of retraining. The models in the clustered ensembles concerned the newspaper data. Clustering continued until three clusters were found.

predictions based on a weighted average of these representatives can be as good as, and sometimes even better than, predictions based on the full ensemble. We believe that this method provides a useful addition to any method featuring an ensemble of models, such as bootstrapping, sampling of Bayesian posterior distributions or multitask learning. The method is not only valuable in terms of predictive quality, but also on a more abstract level. This improvement was apparent on the newspaper data, where different clusters of models brought out different aspects of the data.

A considerable body of literature exists on the subject of the 'overproduce and choose' paradigm, where in the first step a (too) large ensemble of models is trained, and in the second a selection or combination of these models is made to optimize performance (see [36, 66, 68, 72, 80]). Roli *et al* [72] have implemented a clustering algorithm for classifiers, where distance between two classifiers depends on the probability that both misclassify the same pattern. This concept of 'methodological diversity' plays an important role in most methods in this field; in this chapter we have implemented this concept in the entropy term (Equation 5.3), which makes sure cluster centers are optimally diverse. Note that our method is in fact unsupervised, which makes the clustering part less dependent on the availability of supervised data; if model inputs can be generated artificially, the data set used for clustering can be made arbitrarily large.

In this thesis we have chosen the clustering procedure outlined by [73] to perform model clustering. Alternative clustering procedures can of course be considered and, as shown in Section 5.6.2, may sometimes yield equivalent results. We do however stress the importance of using a 'natural' distance function based on model outputs rather than a more arbitrary distance based on model parameters.

At this point we wish to underline that we do not claim to outperform the above mentioned methods in terms of prediction error. This chapter is meant to show that model clustering, through any desired clustering algorithm, is

able to produce a fitting 'summary' of any large ensemble of models. In this thesis we have taken bootstrapping ensembles for an example, but ensembles obtained from sampling a Bayesian posterior distribution, or indeed from any source, can in theory be summarized just as well. The strength of such summaries lies mainly in the fact that they make qualitative analysis easier than it would be on the full ensemble. We demonstrated this through two examples: in the Boston housing example we used the summaries to analyze the bootstrapping process in terms of bias/variance, in the newspaper example we discovered a division of outlets into 'seasonal' outlets and 'short term' outlets. We feel therefore, that the model clustering method described in this chapter can make a valuable contribution in the field of qualitative data analysis (data mining).

Cadez *et al* [20] have done related work on model clustering. An important difference between our work and [20] is that we cluster trained models, slowly increasing the number of clusters, where in [20] clustering and training are combined into a generative model with a fixed number of clusters.

We addressed the problem of retraining a model with a desired architecture from a cluster center expressed in outputs on a data set. Although we recognize that a risk exists in letting the cluster centers run free in output space, we showed that at least for the databases considered in the present model retraining at the end of the clustering procedure does not lead to the wrong representative models. For cases where this method may not be correct, we have suggested saver, yet more time consuming methods where the models are retrained at clearly marked points in the process.

In this thesis we have applied model clustering to gain better predictions and, for the newspaper project, to detect hidden structure in the data. Other applications can be found e.g. in the comparison of networks which do not have the same structure, but are trained to perform the same task. If the network outputs depend strongly on the network's structure, different models are likely to be assigned to different cluster centers. If however two differently structured networks produce similar outputs, there will be clusters inhabited by both types of networks.

Our model clustering method may also have useful applications for (Bayesian) sampling: in this case, an ensemble of models is obtained through sampling from a probability density function which cannot (easily) be expressed analytically. Model clustering can be used to find clusters in this ensemble of samples, and make subsequent analysis easier.

Another application would be the detection of symmetries in a network [76] through study of the differences between clustering based on a distance function dependent on the outputs of the networks and clustering in weight space directly (e.g. with a Euclidean distance function).

Publications

International refereed journals

- B.J. Bakker, T. Heskes, J. Neijt and B. Kappen. Improving Cox survival analysis with a neural-Bayesian approach. *Statistics in Medicine*, In Press, 2002
- B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, 4:83-99
- B. Bakker and T. Heskes. The dynamic hierarchical model with time dependencies at lower levels. *Journal of Time Series Analysis*, Submitted, 2002
- B. Bakker and T. Heskes. Clustering ensembles of neural network models. *Neural Networks*, 16(2):261–269

Conference Proceedings

- B.J. Bakker and T. Heskes. Model clustering by deterministic annealing. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, pages 87–92, 1999
- B.J. Bakker and T. Heskes. A neural-Bayesian approach to survival analysis. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, pages 832–837, 1999
- B.J. Bakker and T. Heskes. Survival analysis: A neural-Bayesian approach. In *Proceedings of the first international conference on Artificial Neural Networks in Medicine and Biology*, pages 162–167, 2000
- B.J. Bakker and T. Heskes. Model Clustering for Neural Network Ensembles. In *Proceedings of the Twelfth International Conference on Artificial Neural Networks*, pages 383–388, 2002

Bibliography

[1] O. Aguilar and M. West. Analysis of hospital quality monitors using hierarchical time series models. In *Case Studies Bayesian Statistics in Science and Technology: Case Studies 4*, New York, 1998. Springer-Verlag.

[2] M. Aitkin and N. Longford. Statistical modelling issues in school effectiveness studies. *Journal of the Royal Statistical Society A*, 149:1–43, 1986.

[3] D. Altman and P.K. Andersen. Bootstrap investigation of the stability of a Cox regression model. *Statistics in Medicine*, 8:771–783, 1989.

[4] V. Arora, P. Lahiri, and K. Mukherjee. Empirical Bayes estimation of finite population means from complex surveys. *Journal of the American Statistical Association*, 92:1555–1562, 1997.

[5] D. Barber and C. Bishop. Ensemble learning for multi-layer networks. In *Advances in Neural Information Processing Systems 10*, pages 395–401, Cambridge, 1997. MIT Press.

[6] D. Barber and B. Schottky. Radial Basis Functions: a Bayesian treatment. In *Advances in Neural Information Processing Systems 10*, pages 402–408, Cambridge, 1997. MIT Press.

[7] J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39, 1997.

[8] J. Berger and M. Delampady. Testing precise hypotheses. *Statistical Science*, 2:317–352, 1987.

[9] E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in Medicine*, 17:1169–1186, 1998.

[10] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford: Clarendon Press, 1995.

[11] J. Bjornstad and R. Butler. The equivalence of backward elimination and multiple comparisons. *Journal of the American Statistical Association*, 83:136–144, 1988.

[12] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 33–42, San Francisco, 1998. Morgan Kaufmann.

[13] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[14] L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24:2350–2383, 1996.

[15] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26, 1998.

[16] B. Brumback and J. Rice. Smoothing spline models for the analysis of nested and crossed samples of curves. *Journal of the American Statistical Association*, 93:961–976, 1998.

[17] S. Bryk and W. Raudenbush. *Hierarchical linear models: applications and data analysis methods.* Sage Publications, Inc, Newbury Park (CAL), 1992.

[18] J. Buhmann and H. Kühnel. Vector quantization with complexity costs. *IEEE Transactions on Information Theory*, 39(4):1133–1145, 1993.

[19] D.W. Bunn and A.I. Vassilopoulos. Comparison of seasonal estimation methods in multi-item short-term forecasting. *International Journal of Forecasting*, 15:431–443, 1999.

[20] I. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *Proceedings of the ACM SIGKDD Conference*, pages 140–149, 2000.

[21] E.A. Camargo and D. Gamerman. Discrete mixture alternatives to dynamic hierarchical models. *Estadstica*, 52:39–77, 2000.

[22] C. Cargnoni, P. Müller, and M. West. Bayesian forecasting of multinomial time series through conditionally Gaussian dynamic models. *Journal of the American Statistical Association*, 92:640–647, 1997.

[23] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[24] R. Caruana, S. Lawrence, and C. Lee Giles. Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13*, pages 402–408, Denver, Colorado, 2001. MIT Press.

[25] D. Cox and D. Oakes. *Analysis of Survival Data.* Chapman Hall, London, 1984.

[26] M. Daniels and C. Gatsonis. Hierarchical generalized linear models in the analysis of variations in health care utilization. *Journal of the American Statistical Association*, 94:29–38, 1999.

[27] M. De Laurentiis and P.M. Ravdin. A technique for using neural network analysis to perform survival analysis of censored data. *Cancer Letters*, 77:127–138, 1994.

[28] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[29] P. Domingos. Why does bagging work? A Bayesian account and its implications. *Proceedings of the third international conference on Knowledge Discovery and Data Mining*, pages 155–158, 1997.

[30] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman & Hall, London, 1993.

[31] D. Gamerman and H. Migon. Dynamic hierarchical models. *Journal of the Royal Statistical Society, Ser. B*, 55:629–642, 1993.

[32] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 472–478. MIT Press, 1996.

[33] M. Goldstein and A. Smith. Ridge-type estimators for regression analysis. *Journal of the Royal Statistical Society*, 36:284–291, 1974.

[34] D. Harrison and D.L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management*, 5:81–102, 1978.

[35] P. Harrison and C.F. Stevens. Bayesian forecasting. *Journal of the Royal Statistical Society B*, 38:205–227, 1976.

[36] S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1996.

[37] B. Hassibi and D. Stork. Second order derivatives for network pruning: optimal brain surgeon. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 164–171, San Mateo, 1993. Morgan Kaufmann.

[38] J. Herndon and F. Harrell. The restricted cubic spline as baseline hazard in the proportional hazards model with step-function time-dependent covariates. *Statistics in Medicine*, 14:2119–2129, 1995.

[39] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.

[40] T. Heskes. Solving a huge number of similar tasks: a combination of multi-task learning and hierarchical Bayesian modeling. In *ICML*, pages 233–241, 1998.

[41] T. Heskes. Empirical Bayes for learning to learn. In P. Langley, editor, *Proceedings of ICML*, pages 367–374, San Francisco, CA, 2000. Morgan Kaufmann.

[42] T. Heskes and O. Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In A. Darwiche and N. Friedman, editors, *Proceedings UAI-2002*, pages 216–233, 2002.

[43] K. Hess. Assessing time-by-covariate interactions in proportional hazards regression models using cubic spline functions. *Statistics in Medicine*, 13:1045–1062, 1994.

[44] G. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the 6th Annual Workshop on Computational Learning Theory*, pages 5–13, New York, 1993.

[45] J. van Houwelingen. Predictability of the survival of patients with advanced ovarian cancer. *Journal of Clinical Oncology*, 7:769–773, 1989.

[46] N. Intrator and C. Kooperberg. Trees and splines in survival analysis. *Statistical Methods in Medical Research*, 4:237–261, 1995.

[47] T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.

[48] E. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

[49] A. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, New York, 1970.

[50] H. Jeffreys. *Theory of Probability.* Oxford University Press, London, third edition, 1967.

[51] W. Jiang and M. Tanner. On the approximation rate of hierarchical mixtures-of-experts for generalized linear models. *Neural Computation*, 11:1183–1198, 1999.

[52] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[53] C. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.

[54] P. van de Laar and T. Heskes. Pruning using parameter and neuronal metrics. *Neural Computation*, 11(4), 1999.

[55] K. Liestol, P.K. Andersen, and U. Andersen. Survival analysis and neural nets. *Statistics in Medicine*, 13:1189–1200, 1994.

[56] X. Lin and D. Zhang. Inference in generalized additive mixed models by using smoothing splines. *Journal of the Royal Statistical Society*, 61:381–400, 1999.

[57] D.V. Lindley and A.F.M. Smith. Bayes estimates for the linear model. *Journal of the Royal Statistical Society B*, 34:1–41, 1972.

[58] G.F. Luger and W.A. Stubblefield. *Artificial intelligence and the design of expert systems.* The Benjamin-Cummings Publishing Company, Redwood City, California, 1989.

[59] D. MacKay. Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. *Network*, 6:469–505, 1995.

[60] D. MacKay and M. Jordan(ed.). *Learning in Graphical Models.* Kluwer Academic Publishers, The Netherlands, 1998.

[61] H. Migon and D. Gamerman. *Statistical Inference: an Integrated Approach.* Arnold, London, 1999.

[62] D. Miller and K. Rose. Hierarchical, unsupervised learning with growing via phase transitions. *Neural Computation*, 8:425–450, 1996.

[63] T. Minka. Expectation propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 362–369, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[64] P. Mortimore, P. Sammons, L. Stoll, D. Lewis, and R. Ecob. *School Matters.* Wells: Open Books, 1988.

[65] R. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag, New York, 1996.

[66] D. Partridge and W.B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4):869–893, 1996.

[67] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2000.

[68] M. Perrone and L. Cooper. When networks disagree: ensemble methods for hybrid neural networks. In R. Mammone, editor, *Neural networks for speech and image processing*, chapter 10, pages 126–142. Chapman-Hall, 1993.

[69] L. Pratt. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems 5*, pages 204–211, 1992.

[70] R. Ripley, A. Harris, and L. Tarassenko. Neural network models for breast cancer prognosis. *Neural Computing & Applications*, 7:367–375, 1998.

[71] C. Robert. *The Bayesian Choice: A Decision-Theoretic Motivation.* Springer, New York, 1994.

[72] F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. In *Multiple Classifier Systems*, pages 78–87, 2001.

[73] K. Rose, E. Gurewitz, and G. Fox. Statistical mechanics of phase transitions in clustering. *Physical Review Letters*, 65:945–948, 1990.

[74] F. Rosenblatt. *Principles of Neurodynamics.* Spartan, New York, 1962.

[75] Donald B. Rubin. EM and beyond. *Psychometrika*, 56(2):241–254, 1991.

[76] S. Rüger and A. Ossen. Clustering in weight space of feedforward nets. In C. Von der Malsburg, editor, *ICANN'96*, pages 83–88, Berlin, 1996. Springer.

[77] P. Rusmevichientong and B. Van Roy. An analysis of belief propagation on the turbo decoding graph with gaussian densities. *IEEE Transactions on Information Theory*, 47:745–765, 2001.

[78] S. Russell and P. Norvig. *Artificial Intelligence. A modern approach.* Prentice Hall, New Jersey, 1995.

[79] M. Seltzer, H. Wong, and A. Bryk. Bayesian analysis in applications of hierarchical models: issues and methods. *Journal of Educational and Behavioral Statistics*, 21:131–167, 1996.

[80] A. Sharkey, N. Sharkey, U. Gerecke, and G. Chandroth. The "test and select" approach to ensemble combination. In *Multiple Classifier Systems*, pages 30–44, 2000.

[81] W. Street. A neural network model for prognostic prediction. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 540–546, Madison, Wisconsin, 1998.

[82] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In *International Conference on Machine Learning*, pages 489–497, 1996.

[83] P. Verweij and H. van Houwelingen. Penalized likelihood in Cox regression. *Statistics in Medicine*, 13:2427–2436, 1994.

[84] N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, February 2002.

[85] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[86] M. West and J. Harrison, editors. *Bayesian Forecasting and Dynamic Models.* Springer, New York, 1997.

[87] C. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

[88] D. Zhang, X. Lin, J. Raz, and M. Sowers. Semiparametric stochastic mixed models for longtitudinal data. *Journal of the American Statistical Association*, 93:710–719, 1998.

Samenvatting

Artificiële intelligentie (AI) beslaat een breed onderzoeksveld met een groeiende rol in het dagelijks leven. Het begrip AI (zie bijv. [58, 78]) varieert van onderwerpen als de nabootsing van menselijk denken en gedrag, mens-machineinteractie en *data mining* tot aan strategieën met als doel het correct uitvoeren van complexe taken. Dit proefschrift zal zich bezighouden met het laatste. Voorbeelden van dit deel van AI zijn schaakcomputers, expert-systemen bij medische diagnose en stem- of vingerafdrukherkenning in beveiligingssystemen.

Een breed scala aan technieken wordt gebruikt om de ambitieuze doelen van AI te bereiken. Eén van deze technieken staat bekend onder de naam *neurale netwerken* (zie bijv. [39]). Het onderzoek in dit veld was aanvankelijk geïnspireerd op de structuur en het functioneren van het menselijk brein, en wel om meerdere redenen: het brein is flexibel, het kan op basis van voorbeelden 'leren' om complexe problemen op te lossen, het kan onvolledige of zelfs tegenstrijdige informatie op adequate wijze verwerken en het is in staat goed te presteren op een groot aantal complexe taken. Het merendeel van de tot nu toe ontwikkelde neurale netwerken-modellen weerspiegelen een abstracte, simpele versie van het biologische neurale netwerk. Abstracte 'neuronen' worden gemodelleerd als wiskundige 'machientjes' die numerieke invoer ontvangen (analoog aan de elektrische signalen die een biologisch neuron ontvangt) en een numerieke uitvoer (die afhangt van de ontvangen invoer) weer doorgeven. Een voorbeeld van zo'n model is het (meerlaags) perceptron, dat in Sectie 5.7 uitgebreider zal worden omschreven. Rosenblatt [74] en Werbos [85] hebben 'leer'-algoritmes voorgesteld die de parameters van dit soort modellen kunnen veranderen, zodanig dat deze modellen na een zekere tijd een taak naar keuze op de juiste manier kunnen uitvoeren. De veranderingen worden gedreven door voorbeelden van 'gewenst gedrag', doorgaans in de vorm van een modelinvoer en de bijbehorende, correcte, modeluitvoer. Deze vorm van 'leren van voorbeelden', die één van de belangrijke onderdelen van dit proefschrift vormt, kan worden gezien als een nadere specialisatie binnen het veld van de AI.

Het veld van de neurale netwerken heeft zich de laatste jaren in twee verschillende richtingen ontwikkeld. Eén deel van het onderzoek in dit veld houdt

zich bezig met het modelleren van de functionaliteit en de biologische complexiteit van het menselijk brein. De complexiteit van de neuron-modellen neemt snel toe, en onderzoekers houden zich bezig met het inpassen van neurofysiologische en anatomische waarnemingen met betrekking tot de functionaliteit van het neurale netwerk. Hun doel is niet zo zeer het bereiken van een bepaalde functionaliteit, maar meer het begrijpen van menselijke cognitie en gedrag in termen van fysiologische en anatomische (sub)cellulaire eigenschappen. Het overige deel van de neurale netwerken-gemeenschap verlaat haar oorspronkelijke biologische inspiratiebron, en concentreert zich op het vinden van leermethoden die in staat zijn een gewenste functionaliteit te realiseren, gericht op toepassingen. Deze tak binnen de neurale netwerken wordt vaak *machine learning* genoemd. Op dit moment worden *machine learning*-methodes meer en meer beheerst door de *probabilistische benadering*. Deze benadering probeert, op zijn eigen manier, de kloof tussen menselijke en artificiële intelligentie te overbruggen. Een deel van deze kloof is te wijten aan het feit dat het merendeel van de huidige computerprogramma's is ontworpen om in zeer strakke termen te 'denken': de temperatuur is 15.7662 graden Celsius, als het gras nat is en de sproeiers hebben niet aangestaan *moet* het geregend hebben, etc. Het menselijk denkproces verloopt echter bepaald anders: we zeggen dat het ongeveer 15 graden is, dat het vannacht waarschijnlijk geregend heeft, etc. Al lijkt dit misschien niet erg nauwkeurig, het lijkt een cruciale en onvermijdelijke stap te zijn in het omgaan met onvolledige, onnauwkeurige informatie, zoals verderop omschreven zal worden.

Neurale netwerken zijn in staat te leren van een set voorbeelden, en hebben op die manier een belangrijke stap gezet in de richting van het menselijk redeneren. Eén van de nadelen die zij nog steeds hebben ten opzichte van menselijke leerlingen kan worden samengevat door het begrip 'overfitting'. Wanneer een (oefen)set voorbeelden gepresenteerd wordt aan een neuraal netwerk, past het leerproces de parameters van het netwerk zo aan dat het deze specifieke voorbeelden zo goed mogelijk kan reproduceren. De oefenset is doorgaans echter een zeer beperkte selectie van alle mogelijke voorbeelden, en bovendien kan ieder voorbeeld door ruis vertekend zijn (i.e., de voorbeelden zijn gekozen om een gewenst gedrag zo nauwkeurig mogelijk na te bootsen, maar doen dit lang niet altijd perfect). Het resultaat is vaak dat het neurale netwerk (door leren) uitkomt op een zeer precies, uitermate gespecialiseerd model dat uitstekend functioneert op de set van oefenvoorbeelden, maar veel slechter onder nieuwe, nog onbekende omstandigheden. Het menselijk leerproces resulteert echter in een veel minder precies 'model' dat alle 'overduidelijke' aspecten van de voorbeelden oppikt, maar de details negeert. In praktijk zien we dat het 'menselijk model' beter kan generaliseren, en daardoor beter presteert.

Een tweede nadeel van neurale netwerk-modellen is hun beperkte mogelijkheid om bestaande expertise te gebruiken. In het ideale geval zou een

model in staat moeten zijn expertise op het gebied van de uit te voeren taak te gebruiken, en deze kennis uit te breiden middels het leren van nieuwe voorbeelden. Expert-systemen zijn echter vaak niet in staat te leren van nieuwe voorbeelden, terwijl neurale netwerken doorgaans *uitsluitend* van nieuwe voorbeelden leren. Een enkele keer is het mogelijk om expertise te incorporeren in de architectuur van een neuraal netwerk, maar dit is zelden makkelijk. Bovendien is het nog steeds onmogelijk om deze kennis door leren uit te breiden, daar het leerproces de architectuur van een netwerk niet verandert. Dit alles is wederom geen probleem voor een menselijke expert. Neem bijvoorbeeld een student kunstgeschiedenis die leert schilderijen van Van Gogh te herkennen. Aan de ene kant zullen zijn leraren hem voorzien van expertkennis betreft kleurgebruik en penseeltechniek, aan de andere kant zal hij zijn herkenningstaak leren uit ervaring. De student zal zich de instructies van zijn leraren waarschijnlijk niet letterlijk weten te herinneren, noch zal hij een precieze techniek ontwikkelen om een Van Gogh van een Degas te onderscheiden. Toch zullen, op een 'zachte' manier, theoretische kennis en ervaring zich in zijn hoofd aaneensmeden, en hem een adequate 'Van Gogh-herkenner' maken.

Dergelijke voorbeelden vormen een inspiratie voor de volgende stap binnen de artificiële intelligentie. We nemen het concept 'leren' uit het veld van de neurale netwerken, en combineren het met het concept kanstheorie. Dit laatste kan gezien worden als een verbinding tussen het exact redeneren van computers en neurale netwerken, en de minder precieze denkprocessen in het menselijk brein. Kanstheorie is enerzijds exact: het definieert kansverdelingen die goed gedefinieerde functies zijn van een bekende set parameters die met hoge precisie in een computer kunnen worden opgeslagen. Computers kunnen zonder problemen omgaan met kansverdelingen: er bestaan nette regels voor het berekenen van de waarschijnlijkheid van een model gegeven een aantal waarnemingen (die door het model omschreven kunnen worden). Verder bestaan er regels die omschrijven hoe een verdeling die de kans op een waarneming gegeven een stelsel van expertise enerzijds, en een kansverdeling gebaseerd op een serie voorbeelden anderzijds, gecombineerd kunnen worden. Het probabilistische redeneren staat echter één stap dichter bij het inexacte, menselijke redeneren dan redeneren zonder enige onzekerheid. De probabilistische benadering gebruikt nog altijd exacte waarden voor de representatie van gegevens en modelparameters, in de vorm van gemiddeldes van kansverdelingen, maar nu gaan deze waarden gepaard aan uitdrukkingen van hun (on)zekerheid, uitgedrukt in standaarddeviaties en varianties. Dit kan worden gezien als een exacte benadering van de zachte representatie van modellen en kennis in het menselijk brein. Onze hoop (en ervaring) is dat deze nieuwe manier om parameterwaarden uit te drukken enkele nadelen van traditionele AI ten opzichte van menselijk denken zal verwijderen.

Een voorbeeld van het verschil tussen het redeneren zonder onzekerheid en de probabilistische benadering is het volgende. Beschouw de exacte stelling

'de trein van Edinburgh naar Inverness vertrekt om $t_E = 15.30$u'. Vertaald in probabilistische termen wordt dit

$$t_E \sim P(t_E|\hat{t}_E, \sigma_E^2) \,,$$

i.e., t_E wordt beschreven door een kansverdeling met $\hat{t}_E (= 15.30\text{hu})$ als gemiddelde en een standaarddeviatie σ_E. Dit laatste geeft aan hoeveel een werkelijke meting van t_E naar verwachting af zal wijken van $\hat{t}_E$. Als we dagelijks de trein vanaf Edinburgh zouden nemen, zouden we verwachten dat hij gemiddeld genomen om 15.30u vertrekt, maar ook dat het verschil tussen $t = 15.30$u en de werkelijke vertrektijd t_E gemiddeld gelijk is aan σ_E (minuten). Een voorbeeld van de toegevoegde waarde van de probabilistische benadering is het volgende: als een exact model van de Britse spoorwegen zou stellen dat mijn trein vanuit Londen om $t_L = 15.28$u aankomt en dat $t_E = 15.30$h, dan zal het me ook zeggen dat ik de overstap zonder meer haal. De probabilistische benadering zal echter beide tijden in termen van kansverdelingen uitdrukken, en me vertellen dat ik mag verwachten mijn overstap te halen, maar het zal me ook vertellen wat de kans is dat ik hem mis. Gebaseerd op deze extra informatie kan ik vervolgens besluiten dit risico te nemen, of een trein eerder.

Dit proefschrift draagt bij aan één van de actuele ontwikkelingen binnen de AI: de combinatie van het leren van voorbeelden, zoals dat reeds gedaan is binnen neurale netwerken, en het gebruik van kanstheorie. Deze combinatie geeft toegang tot een volledig nieuwe vorm van leren, en maakt een breed scala aan robuuste, flexibele modellen mogelijk, die op vele manieren de klassieke neurale netwerken-modellen overvleugelen. Dit proefschrift richt zich vooral op een deel van kanstheorie dat zeer geschikt is voor leren: de Bayesiaanse benadering, die in meer detail zal worden beschreven in de volgende sectie. Het doel van dit proefschrift is het uitbreiden van bestaande (leer)methoden via deze benadering. Het zal worden gedemonstreerd dat het overfitting probleem op deze manier vaak kan worden voorkomen, en dat de Bayesiaanse benadering een uitstekende manier is om expert-kennis in modellen in te bouwen. Er bestaat een enorme variëteit aan modellen, in diverse toepassingsgebieden, die allemaal kunnen profiteren van een verbetering in de vorm van de Bayesiaanse benadering. Dit proefschrift realiseert zulke verbeteringen op vier verschillende gebieden, die beschreven zullen worden in Sectie 5.7.

Bayesiaans leren

Bayesiaanse statistiek is een deel van de kanstheorie dat de laatste tien jaar aan populariteit heeft gewonnen binnen het veld van de neurale netwerken. Het theorema waaraan dit deel van de statistiek zijn naam ontleent, is het theorema van Bayes:

$$P(M|O) = \frac{P(O|M)}{P(O)} P(M) \, .$$

Laat M een zeker model van de wereld representeren, en laat $P(M)$ het *a priori* vertrouwen in dit model zijn. Het theorema van Bayes zegt ons nu wat te doen als ons vertrouwen in model M op de proef wordt gesteld ofwel, wanneer we een zeker verschijnsel O waarnemen dat kan worden beschreven door ons model. De breuk in bovenstaande formule zal groter zijn dan één wanneer de waarschijnlijkheid van de nieuwe observatie onder het huidige model ($P(O|M)$) groter is dan haar 'gemiddelde' waarschijnlijkheid $P(O)$, en anders kleiner dan één. De gemiddelde waarschijnlijkheid $P(O)$ kan uitgedrukt worden als $P(O) = \int dM \, P(O|M)P(M)$, een gemiddelde over modelvoorspellingen dat gewogen wordt met het vertrouwen in ieder model. Dus, als model M beter in staat is de nieuwe waarneming O te beschrijven dan het gemiddelde model, wordt ons vertrouwen in M gesterkt, anders verzwakt.

Eén doel van Bayesiaans leren is het beoordelen van een model M op grond van zijn vermogen zo veel mogelijk waarnemingen O te kunnen beschrijven (binnen de grenzen van een specifieke leeropdracht). In praktijk is het niet mogelijk iedere denkbare waarneming mee te nemen, dus in plaats hiervan wordt een beperkte set gegevens D gebruikt, die *oefenset* genoemd wordt. De marginale waarschijnlijkheid van de oefenset, $P(D)$, is over het algemeen onbekend en wordt genegeerd binnen het leerproces. Het meer interessante deel van het leren ligt in het vinden van geschikte definities voor $P(D|M)$, ook wel de *data likelihood* genoemd, en $P(M)$, die we de *prior verdeling* noemen. Merk op dat de data likelihood erg verschilt van de marginale waarschijnlijkheid $P(D)$. De eerste definieert de waarschijnlijkheid van een waarneming onder een model M, en is daardoor van groot belang in het waarderen van M's vermogen om waarnemingen te beschrijven. De andere is de kans waarmee een specifieke oefenset D uit alle 'mogelijke' waarnemingen gekozen kan worden, hetgeen geen rol speelt in het leerproces.

In de volledig Bayesiaanse benadering volstaat het uitdrukkingen te vinden voor de prior en de data likelihood. Hiermee kan de *a posteriori waarschijnlijkheid* (of posterior) $P(M|D)$ geconstrueerd worden voor ieder model M. Deze posterior kan vervolgens worden gebruikt om het meest waarschijnlijke model te vinden, of om voorspellingen te genereren voor nieuwe waarnemingen. Zulke waarnemingen komen vaak voor als combinaties van een bekende invoer x en een uitvoer y die voorspeld dient te worden door het model. Om, gegeven x, y te voorspellen zoeken we een uitdrukking

$$\begin{aligned} P(y|x, D) &= \int dM P(y|x, D, M)P(M|x, D) \\ &= \int dM P(y|x, M)P(M|D) \, , \end{aligned}$$

waar we in de tweede regel D en x hebben weggelaten aangezien de waarschijnlijkheid van y niet meer afhankelijk is van de oefendata wanneer het model M een keer gegeven is, en de waarschijnlijkheid van het model M onafhankelijk wordt geacht van nieuwe invoer x. We kunnen nu de verwachtingswaarde voor y onder dit model,

$$\langle y \rangle = \int dy \; yP(y|x, D) \, ,$$

gebruiken als voorspelling.

In de volledig Bayesiaanse benadering stellen we de prior $P(M)$ van te voren vast en laten deze onveranderd gedurende de rest van de procedure. In de *empirisch* Bayesiaanse benadering is het toegestaan deze prior tijdens het optimalisatieproces nog te veranderen. In het volgende vatten we de *hyperparameters* die de prior beschrijven samen als Λ. We zoeken nu die waarden voor Λ die, gegeven de waarnemingen, het meest waarschijnlijk zijn:

$$\begin{aligned} \Lambda_{\text{opt}} &= \underset{\Lambda}{\text{argmax}} \, P(\Lambda|D) \\ &= \underset{\Lambda}{\text{argmax}} \, P(D|\Lambda)\frac{P(\Lambda)}{P(D)} \\ &\propto \underset{\Lambda}{\text{argmax}} \int dM P(D|M)P(M|\Lambda) \, , \end{aligned}$$

waar iedere waarde voor Λ *a priori* even waarschijnlijk wordt geacht (hoewel het in principe mogelijk is om een zogenaamde *hyperprior* verdeling $P(\Lambda)$ te construeren).

Begrippen en toepassingsgebieden

Eén van de mogelijkheden die kanstheorie biedt, is het uitdrukken van het begrip 'soortgelijk' in wiskundige termen. Ieder mens zal dit begrip intensief gebruiken, binnen alle aspecten van het alledaagse redeneren. Doorgaans zien we het als ervaring: wanneer we een probleem tegenkomen dat erg 'lijkt' op een probleem dat we al eerder zijn tegengekomen (en opgelost hebben), proberen we gewoon de zelfde strategie opnieuw toe te passen. Wellicht komen we erachter dat, aangezien het probleem niet exact gelijk is aan hetgeen we eerder opgelost hebben, de strategie enigszins aangepast dient te worden, maar in grote lijnen zal zij nog steeds kloppen. Ook andersom werkt het. Wanneer we regelmatig op elkaar lijkende problemen oplossen ontwerpen we misschien de eerste paar keer steeds weer opnieuw een strategie en vinden het wiel een keer of wat opnieuw uit, maar na verloop van tijd zullen we toch een gemene deler in onze inspanningen ontdekken, en deze 'gemiddelde' strategie gebruiken als uitgangspunt voor iedere volgende poging.

Ook computers kunnen geïnstrueerd worden gebruik te maken van overeenkomstigheden in leertaken. Dit proefschrift zal een aantal situaties behandelen waar de computer een collectie onafhankelijke, maar soortgelijke taken dient uit te voeren. Het feit dat er overeenkomsten bestaan tussen de taken kan gezien worden als een vorm van expertkennis. Deze kennis kan worden uitgedrukt in een Bayesiaanse prior, die vervolgens kan worden gebruikt om collecties van taken beter te modelleren. In dit proefschrift zullen we zowel voorbeelden tonen waar het schatten van zulke priors leidt tot het ontdekken van een gemiddelde strategie, als ook voorbeelden waar de kennis die vergaard wordt bij het leren van één taak gebruikt wordt om andere taken beter te leren.

Survival analysis

Survival analysis is een leerprobleem dat kan worden omschreven als een stel parallelle taken, waarbij we de kans willen weten die een groep patiënten heeft om een zekere periode te overleven na de aanvankelijke diagnose van een ernstige, levensbedreigende ziekte. Men kan vragen met welke waarschijnlijkheid men een periode van vijf maanden overleeft, of twee jaar, of drie jaar. Bestaande methoden schatten deze waarschijnlijkheden door observatie van patiënten die langer dan vijf maanden overleven, of twee jaar, of drie jaar. Vaak echter werden deze vragen benaderd via afzonderlijke wegen, daar men er geen rekening mee hield dat de structuren en gegevens die onder deze taken schuilgaan, soortgelijk zijn, vooral wanneer voorspellingen gedaan worden voor periodes die dicht bij elkaar liggen. Dientengevolge was het mogelijk dat erg onnauwkeurige voorspellingen gedaan werden. In Hoofdstuk 2 wordt een priorverdeling geïntroduceerd die gebruik maakt van het besef dat voorspellingen voor ongeveer even lange perioden niet al te zeer zouden moeten verschillen. Een tweede priorverdeling zal worden ontworpen om te voorkomen dat voorspellingen voor verschillende patiënten met soortgelijke symptomen niet *te* zeer verschillen. Dit voorkomt overfitten op de oefendata, hetgeen een belangrijk probleem is in het klassieke model.

Het clusteren van taken

Een ander voorbeeld is het voorspellen van de losse verkoop van kranten op een aantal verschillende verkooppunten. Hoewel iedere locatie zijn eigen karakteristieken zal hebben, kan men er van uitgaan dat de voorspeltaken ook voor verschillende verkooppunten wel soortgelijk zijn. We coderen deze overtuiging in een Bayesiaanse priorverdeling, en ook in de modelstructuur van ieder voorspellingsmodel. Ofwel: één deel binnen alle modellen is hetzelfde voor ieder model en geeft hun algemene, gedeelde eigenschappen weer, het andere deel 'lijkt' slechts op de corresponderende delen bij de andere modellen, en

biedt ruimte aan verschillen die te maken hebben met de specifieke verkooppunten. Voor dit tweede deel wordt de gelijkheidsaanname geïmplementeerd door (een deel van) ieder model te onderwerpen aan een gemeenschappelijke prior, gebaseerd op een 'gemiddeld model'. Op deze manier is het slechts nodig steeds de overeenkomsten tussen een model voor één specifiek verkooppunt en het gemiddelde model te beschouwen, en hoeven we geen paarsgewijze vergelijkingen te maken tussen iedere mogelijke combinatie van twee modellen.

De gecreëerde 'link' tussen soortgelijke modellen leidt tot een sterker generaliserend vermogen voor iedere model. Hoewel verschillende modellen toegestaan worden voor verschillende taken, kan ieder model gebruik maken van de volledige set oefendata, van alle taken. Daar ieder model op deze manier geoptimaliseerd wordt op basis van de volledige database in plaats van de (veel kleinere) taak-specifieke dataset, wordt het risico van overfitten veel kleiner. Men zou ook kunnen zeggen dat de modellen 'van elkaar leren'. Vanuit die optiek dient de link tussen modellen als een doorgeefluik om kennis die vergaard is door één model door te geven aan een ander.

Soms is de informatie beschikbaar dat sommige verkooppunten hoogstwaarschijnlijk meer op elkaar lijken dan op andere modellen: krantenverkoop in badplaatsen en andere recreatiegebieden zou zich best eens op een andere manier kunnen gedragen dan de verkoop in grote steden. Dit kan uitgedrukt worden in Bayesiaanse priors. In plaats van één gemiddeld model, staan we meerdere 'lokaal gemiddelde' modellen toe. Voor ieder verkooppunt wordt het model 'toegewezen' aan het gemiddelde model waar het het meest 'op lijkt', samen met die modellen waar het de grootste overeenkomsten mee vertoont. De modellen zijn op deze manier niet alleen beter in staat hun voorspeltaak uit te voeren: door het bestuderen van de gevormde 'taakclusters' kunnen wij ook meer leren over de structuur van de taken.

Het hiërarchische dynamische model

Survival analysis bevat een relatie tussen voorspellingen voor naburige tijden. Hetzelfde kan worden gedaan voor de krantenverkoop, omdat de verkoopcijfers van deze week waarschijnlijk wat zeggen over de verkopen van volgende week. In dit geval passen we een idee uit de wereld van de tijdreeksenanalyse toe: een zogenaamd dynamisch lineair model. We nemen aan dat de verkoop op een willekeurige locatie een functie is van een zekere 'verborgen toestand', die zich door de tijd heen ontwikkelt. In dit model zijn alle toestanden naar verwachting soortgelijk, maar niet identiek, aan de vorige toestand (de toestand die gerelateerd is aan dezelfde locatie, maar dan één tijdstap eerder), of een bekende transformatie van die toestand. Los daarvan houden we vast aan het idee dat de verkoop op alle locaties zich min of meer hetzelfde gedraagt. Verborgen toestanden voor verschillende verkooppunten, maar voor het zelfde tijdstip, zijn verbonden met één 'gemiddelde toestand'. Iedere toestand wordt

verondersteld getrokken te zijn uit een kansverdeling die zowel afhangt van de vorige toestand, die verschillend is voor ieder verkooppunt, als van de gemeenschappelijke factor die geïmplementeerd wordt als de gemiddelde toestand.

Het clusteren van ensembles

Wanneer taken niet van nature voorkomen als een ensemble soortgelijke taken, kunnen we ze nog altijd hiertoe aanpassen. In het veld van de neurale netwerken bestaan vele methoden die een ensemble van soortgelijke modellen opleveren. Eén van die methoden wordt *bootstrapping* genoemd (zie bijv. [30]): in plaats van het trainen van één model op één volledige database, wordt een ensemble van modellen getraind, op licht van elkaar verschillende subsets van de volledige database. Op deze manier representeren we een taak (nl. leren op de volledige dataset) als een collectie van soortgelijke taken (nl. leren op een groot aantal subsets van de volledige dataset). Een ander voorbeeld ligt besloten in de Bayesiaanse benadering van leren zelf. Hier leren we niet één model, maar stellen een kansverdeling op over oneindig veel mogelijke modellen. Het trekken van samples uit deze verdeling (zie Sectie 1.3.3) levert een grote (doch eindige) verzameling vergelijkbare modellen op, zelfs wanneer het niet mogelijk is expliciet een collectie soortgelijke taken te specificeren.

Ensembles van modellen vormen vaak een accurate, evenwichtige benadering van een leertaak, maar het is vaak wel moeilijk om een goed overzicht te houden over de karakteristieken van al deze modellen. Het zou makkelijker zijn als er een soort compacte samenvatting beschikbaar was die alle modelkarakteristieken in een hanteerbare vorm kon samenvatten. In Hoofdstuk 5 zullen we een dergelijke samenvatting presenteren, in de vorm van *clustercentra*. De modelensembles worden verdeeld in clusters, waarbij modellen binnen één cluster meer op elkaar 'lijken' dan modellen in twee verschillende clusters. Ieder cluster wordt gerepresenteerd door een 'representatief model', of clustercentrum. We zullen aantonen dat een representatie van een volledig ensemble van modellen door een klein aantal clustercentra het ensemble meer hanteerbaar maakt. Dit gereduceerde ensemble van clustercentra is echter nog steeds even goed in staat de gewenste waarnemingen te beschrijven als het volledig ensemble.

Modellen, methoden en benaderingen

Iedere hoofdstuk in dit proefschrift beschrijft een model, gekozen om een specifieke taak uit te voeren, en een posteriorverdeling. De laatste dicht een waarschijnlijkheid toe aan ieder mogelijk model, gebaseerd op de gegevens in een oefenset. Het model, en daarmee de posteriorverdeling, hebben een verschillende vorm in ieder hoofdstuk. De posterior heeft vaak zo'n complexe

structuur dat zij niet makkelijk uitgedrukt kan worden als functie van de hyperparameters. Er is doorgaans een hoog-dimensionale integratie bij betrokken die niet exact uitgevoerd kan worden, ofwel, er bestaat geen analytische uitdrukking voor het resultaat van de integratie. In zulke gevallen moeten we onze toevlucht nemen tot benaderingen van de exacte verdeling. Het ontwerp en de implementatie van zulke benaderingen beslaan in feite het grootste deel van het werk dat in dit proefschrift beschreven wordt. Het overige deel van deze sectie geeft een beschrijving van een breed scala van modellen en benaderingsmethoden, en beschrijft hoe deze geïmplementeerd worden in dit proefschrift.

Het meerlaags perceptron

In de loop van dit proefschrift zal vaak een modelstructuur genaamd 'meerlaags perceptron' (MLP) gebruikt worden om voorspellingen te doen voor waarneembare uitvoer op basis van een gegeven invoer. De architectuur van zo'n model is weergegeven in Figuur 8.1. De cirkels onder in het plaatje representeren de invoer van het model. De invoer is verbonden met de middelste laag cirkels, die we 'hidden units' noemen, aangezien ze corresponderen met in het model gebruikte waarden die geen waarneembare in- of uitvoer representeren. Iedere hidden unit is verbonden met alle invoereenheden, wat aangeeft dat zijn waarde afhangt van een gewogen som van deze eenheden:

$$h_i = g\left(\sum_j W_{ij} x_j + b_1\right) ,$$

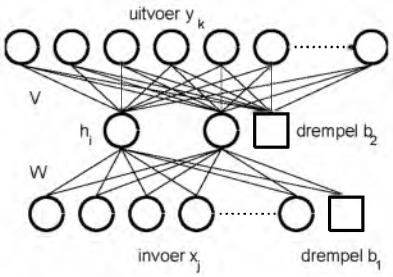
waarbij de 'gewichten' W_{ij} gerepresenteerd worden door de verbindingslijntjes, en 'invoer-naar-hidden'-gewichten worden genoemd. De parameters h_i representeren de waarden van de hidden units, b_1 representeert de 'drempel' (een toegevoegde constante) en de parameters x_j representeren de invoer van het model.

De bovenste laag cirkels representeert de uitvoer van het model, die afhankelijk is van een gewogen som van de waarden van de hidden units:

$$y_k = f\left(\sum_i V_{ki} h_i + b_2\right) ,$$

waarbij de parameters y_k de modeluitvoer representeren en de parameters V_{ki} de 'hidden-naar-uitvoer'-gewichten.

In een lineair MLP doen de transformatiefuncties $g(.)$ en $f(.)$ 'niets': $g(z) = f(z) = z$. In veel gevallen zullen we echter een niet-lineaire MLP met niet-lineaire transformatiefuncties, zoals de tangens hyperbolicus gebruiken.



Figuur 5.6: Het meerlaags perceptron.

Niet-lineaire MLPs hebben een invoer-uitvoer-verband dat complexer is dan bij lineaire MLPs, en kunnen daardoor ook complexere functies modelleren.

Het aantal hidden units zal, voor de in dit proefschrift gebruikte MLPs, kleiner zijn dan het aantal invoereenheden. Op deze manier wordt een lagerdimensionale weergave van de invoereenheden gerealiseerd. Dit leidt tot de automatische detectie van (belangrijke) kenmerken van de gebruikte gegevens, en tot een betere generalisatie [7, 23].

Het proportional hazards model

Het model dat binnen survival analyse het meest gebruikt wordt is het proportional hazards model. Dit model drukt de waarschijnlijkheid dat een patiënt tot een bepaalde tijd overleeft uit als een eenvoudig product van twee functies. De eerste functie, proportional hazard genaamd, hangt af van de invoer die de patiënt beschrijft (leeftijd, gewicht, etc.), maar niet van de tijd. De tweede functie, de zogenaamde baseline hazard, hangt alleen af van de tijd en is daarmee hetzelfde voor alle patiënten.

De baseline hazard wordt vanouds als volgt geschat. De tijd wordt verdeeld in een aantal discrete intervallen. De geschatte baseline hazard voor ieder interval wordt berekend als het aantal patiënten (dat gevolgd wordt in het onderliggend medisch onderzoek) dat overlijdt binnen dit interval, gedeeld door het aantal patiënten dat in leven was aan het begin van dit interval. Toevallige fluctuaties kunnen hierbij leiden tot een erg 'wilde' variatie in de baseline hazard door de tijd heen, zeker wanneer het aantal bestudeerde patiënten erg beperkt is.

In Hoofdstuk 2 implementeren we een variant van de proportional hazards methode in de vorm van een niet-lineair MLP. We leggen twee priorverdelingen op aan de modelparameters. Eén prior dient om te voorkomen dat de baseline hazard al te wild wordt. De andere prior werkt op de proportional hazard, en dient (ook) om overfitten op de oefendata te voorkomen.

Het trekken van samples uit de posterior

Een kansverdeling kan worden gerepresenteerd door een collectie samples. Neem bijvoorbeeld de verdeling $P(M|\Lambda)$ over mogelijke modellen M. We kunnen hieruit een aantal verschillende modellen M_i selecteren. Wanneer de kans dat we een specifieke M_i kiezen gegeven wordt door $P(M_i|\Lambda)$, kan men zeggen dat we *samples trekken* uit $P(M|\Lambda)$. Het trekken van samples kan bijvoorbeeld worden gebruikt om een integraal te benaderen: wanneer we een (groot) aantal kandidaatmodellen M_i (gedefinieerd door hun modelparameters) uit $P(M|\Lambda)$ hebben getrokken, kunnen we $P(\Lambda|O) \propto \int dM P(O|M)P(M|\Lambda)$ benaderen door $P(O|M_i)$ uit te rekenen voor ieder getrokken sample, en hiervan het gemiddelde te nemen. Deze benadering is een populaire vorm van *numerieke integratie.*

Uit sommige kansverdelingen (bijv. Gaussisch of uniform) kan men erg makkelijk samples trekken. De verdelingen die veel voorkomen binnen het Bayesiaans leren, zijn vaak echter ingewikkelder, en vereisen complexere procedures. Eén zo'n procedure is *importance sampling.* Hierbij trekken we samples uit een standaardverdeling (bijv. Gaussisch) die kan lijken op het complexere origineel. Iedere trekking wordt geaccepteerd als zijn waarschijnlijkheid onder de complexe verdeling ($P(s|C)$) groter is dan zijn waarschijnlijkheid onder het alternatief ($P(s|G)$). Zo niet, dan wordt hij met kans $P(s|C)/P(s|G)$ geaccepteerd en anders verworpen. In praktijk is het vaak erg moeilijk om een standaardfunctie $P(s|G)$ te vinden die erg lijkt op de verdeling waaruit we samples willen trekken. Het resultaat is dat veel samples een erg lage verhouding $P(s|C)/P(s|G)$ zullen hebben. Zulke samples zullen hoogstwaarschijnlijk verworpen worden, en zijn dan voor niets getrokken. Markov chain Monte Carlo (MCMC) sampling, dat dit nadeel niet heeft, zal in Hoofdstuk 2 worden besproken.

Benaderende verdelingen

Samples trekken van een posteriorverdeling is altijd mogelijk en levert, in de limiet van oneindig veel samples, exacte resultaten voor alle verwachtingswaarden. Er is echter wel een boel tijd nodig om genoeg samples te verzamelen, en verdere berekeningen kunnen eleganter uitgevoerd worden als de daadwerkelijke posteriorverdeling als een expliciete functie aanwezig is. Wanneer deze verdeling te complex is om direct te gebruiken, nemen we vaak genoegen met een benaderende verdeling. Deze verdeling zal doorgaans een simpeler vorm hebben (bijv. Gaussisch), en kan makkelijk gebruikt worden voor iedere verdere berekening. De benadering moet natuurlijk zo veel mogelijk lijken op

de exacte verdeling. Een veelgebruikte maat voor hoeveel twee verdelingen op elkaar lijken, is de Kullback-Leibler (KL)-divergentie:

$$\mathrm{KL}(P(.), Q(.)) = \int dx P(x) \frac{\log P(x)}{\log Q(x)} ,$$

waarbij $P(.)$ en $Q(.)$ de twee verdelingen zijn en x de parameter is waarvan de waarschijnlijkheid uitgedrukt wordt. De KL-divergentie is nul wanneer $P = Q$ en anders positief. De optimale benadering volgt uit minimalisatie van haar (KL)-divergentie met de exacte posteriorverdeling.

Een veel simpeler manier om een benaderende verdeling te schatten is de Laplace-benadering. De benaderende verdeling $P(x)$ wordt in dit geval verondersteld Gaussisch te zijn, met een gemiddelde $\bar{x}$ die de modus is van de exacte verdeling. De covariantie is gerelateerd aan de tweede afgeleide (naar x) van de logaritme van de exacte verdeling, genomen in het maximum:

$$\sigma^{-2} = \left[-\frac{\partial^2}{\partial x^2} \log P(x) \right]_{x=\bar{x}} .$$

Deze methode is vaak niet erg nauwkeurig, doch vereist soms veel minder rekentijd.

De posteriorverdeling $P(x)$ in Hoofdstuk 2 is te complex om direct gebruikt te worden. We stellen derhalve een variationele methode voor waarbij we de posterior benaderen met een simpele verdeling $Q(x)$, en we de KL-divergentie $\mathrm{KL}(P(x), Q(x))$ minimaliseren. We vergelijken deze benadering met MCMC sampling en met de Laplace-benadering. We berekenen verwachtingswaardes voor de modeluitvoer onder iedere benadering, en vergelijken de drie methodes in termen van nauwkeurigheid van de voorspellingen.

Invoeranalyse en de Bayesfactor

Eén van de taken van ieder model in dit proefschrift is het voorspellen van nieuwe, nog onwaargenomen uitvoer te voorspellen op basis van een nieuwe, waargenomen invoer. Een deel van deze invoer speelt wellicht een cruciale rol bij deze voorspelling, terwijl een ander deel misschien wel volledig ongerelateerd is aan de uitvoer. Dat deel van de invoer dat niet bijdraagt aan de juiste voorspelling van nieuwe uitvoer zou het leerproces best eens kunnen storen: het 'verwart' het model, en maakt het moeilijker om de 'werkelijke' afhankelijkheid van het andere deel van de invoer te vinden. Het kan derhalve belangrijk zijn, zeker als er veel verschillende soorten invoer aanwezig zijn, om bruikbare invoer te onderscheiden van 'nutteloze invoer'. Een elegante methode om het model te ontdoen van irrelevante invoer is gerelateerd aan de *Bayesfactor*. Deze factor is gedefinieerd als

$$BF = \frac{P(D|H_0)}{P(D|H_1)} ,$$

de verhouding van de waarschijnlijkheid van de (oefen)data gegeven een zekere nulhypothese en de waarschijnlijkheid gegeven een andere hypothese. Om het belang van een zekere invoer vast te stellen, definiëren we de nulhypothese als: de invoer is irrelevant, en kan verwijderd worden uit het model. De alternatieve hypothese is het tegenovergestelde: de invoer moet blijven. Een hoge Bayesfactor geeft aldus een sterke indicatie dat de beschouwde invoer irrelevant is.

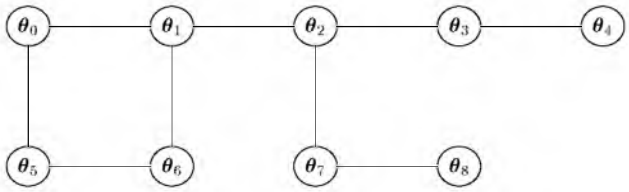
In Hoofdstuk 2 gebruiken we de Bayesfactor om af te komen van alle 'nutteloze invoer', en laten we zien dat dit een positieve invloed heeft op de prestaties van het model.

Multitask learning

'Multitask learning' is een concept dat gebruik maakt van het feit dat modellen die getraind worden voor soortgelijke taken ook op elkaar moeten lijken. Dit kan zowel worden geïmplementeerd door het 'hard delen' en het 'zacht delen' van parameters. Hard gedeelde parameters maken deel uit van de modelstructuur zelf. Het meerlaags perceptron heeft een geschikte vorm om dit te implementeren. Laat ieder van de uitvoereenheden van het MLP de uitvoer voor één taak representeren, en laat het aantal hidden units veel kleiner zijn dan het aantal uitvoereenheden (zie ook Figuur 6.1). Het is makkelijk te zien dat, hoewel iedere uitvoereenheid (taak) zijn eigen set hidden-naar-uitvoer gewichten heeft, alle taken dezelfde invoer-naar-hidden-gewichten delen. In principe gebruiken alle taken dezelfde lager-dimensionale representatie van de invoer, maar hebben ze hun eigen uiteindelijke transformatie van deze representatie.

Zacht delen kan worden geïmplementeerd door middel van een gemeenschappelijke priorverdeling over (een gedeelte van) de parameters van ieder model. Deze priorverdeling kan een enkele Gaussische verdeling zijn, wat impliceert dat alle zacht gedeelde parameters verdeeld zijn rond dezelfde gemiddelde parameterset, of hij kan complexer van aard zijn. Een som van Gaussische verdelingen kan bijvoorbeeld worden gebruikt om meerdere taakclusters toe te staan. Ieder cluster wordt dan gekarakteriseerd door het gemiddelde en de variantie van een van de Gaussische verdelingen in de som. Voor iedere taak worden de waarschijnlijkheden bepaald om bij ieder van de clusters te horen, die deels afhankelijk zullen zijn van de waarnemingen die bij die taak horen. Dit clusteren van taken leidt tot meer accurate voorspellingen van toekomstige waarnemingen, en biedt een nieuw inzicht in de karakteristieken van iedere taak.

In Hoofdstuk 3 implementeren we zowel het hard als het zacht delen van parameters, en bestuderen we de effecten van zowel de enkele Gaussische prior als van clusteren.



Figuur 5.7: Voorbeeld van een grafisch model. De open cirkels representeren toestanden, de verbindingen geven voorwaardelijke onafhankelijkheden aan.

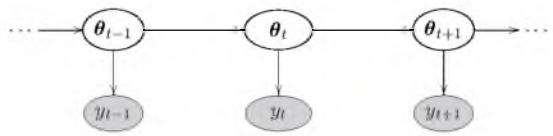
Grafische modellen

Grafische modellen worden vaak gebruikt om complexe systemen van interagerende variabelen te representeren. Beschouw het model dat afgebeeld is in Figuur 6.2. De open cirkels in dit plaatje representeren de toestanden van dit model. Toestanden zijn vectoren die lokale delen van het gemodelleerde systeem beschrijven. Sommige toestanden zijn met elkaar verbonden, wat wijst op voorwaardelijke onafhankelijkheden. Neem bijvoorbeeld de drie toestanden $\boldsymbol{\theta}_2$, $\boldsymbol{\theta}_3$ en $\boldsymbol{\theta}_4$ in Figuur 6.2. De toestanden $\boldsymbol{\theta}_2$ en $\boldsymbol{\theta}_3$ zijn verbonden in het plaatje, net als de toestanden $\boldsymbol{\theta}_3$ en $\boldsymbol{\theta}_4$, maar $\boldsymbol{\theta}_2$ en $\boldsymbol{\theta}_4$ niet. Dit betekent dat als $\boldsymbol{\theta}_3$ bekend is, de waarschijnlijkheid van $\boldsymbol{\theta}_4$ onafhankelijk is van $\boldsymbol{\theta}_2$, ofwel $P(\boldsymbol{\theta}_4|\boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = P(\boldsymbol{\theta}_4|\boldsymbol{\theta}_3)$. Zulke onafhankelijkheden maken het mogelijk om relatief makkelijke uitdrukkingen op te schrijven voor $P(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N)$, de gezamenlijke waarschijnlijkheid van alle toestanden in het netwerk.

Tijdreeksen worden vaak gerepresenteerd als gerichte grafische modellen. In een *gerichte* graaf (Figuur 6.3) zijn de knopen verbonden via pijlen. Wanneer bijvoorbeeld een pijl van toestand $\boldsymbol{\theta}_t$ naar $\boldsymbol{\theta}_{t+1}$ wijst, betekent dit dat er een uitdrukking bestaat voor $P(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$, de waarschijnlijkheid van de toekomstige toestand gegeven de huidige toestand (maar niet noodzakelijkerwijs voor de gezamenlijke waarschijnlijkheid $P(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1})$). Een grafisch model dat zulke onafhankelijkheden vertoont heet een *Markoviaanse* dynamica te vertonen, ofwel: de toestand van het systeem op tijd t ($\boldsymbol{\theta}_t$) hangt alleen af van de voorgaande toestand, $\boldsymbol{\theta}_{t-1}$. De toestanden kunnen verbonden zijn met waarnemingen, gerepresenteerd door ingekleurde ellipsen. De verbinding geeft, net als eerder, aan dat de waarschijnlijkheden van deze waarnemingen (op dezelfde tijd t) afhangen van de toestanden waarmee ze verbonden zijn. Dit type grafisch model wordt vaak gebruikt om tijdreeksen te beschrijven.

Het Kalmanfilter

Wanneer de toestanden van een grafisch model continue variabelen hebben, wordt hun afhankelijkheid van de voorgaande toestand vaak gemodelleerd mid-



Figuur 5.8: Voorbeeld van een gericht grafisch model. De ingekleurde gebieden representeren waarnemingen, de open ellipsen representeren toestanden.

dels een *Kalmanfilter*. Dit betekent dat de voorspelling van de huidige toestand als volgt luidt:

$$\boldsymbol{\theta}_t = A\boldsymbol{\theta}_{t-1} + \boldsymbol{\nu}_t \,,$$

een lineaire functie van de voorgaande toestand (via de *evolutiematrix* A) met toegevoegde Gaussische ruis $\boldsymbol{\nu}_t$. Een tijdreeksmodel dat bepaald wordt door dit type evolutievergelijking wordt een dynamisch lineair model (DLM) genoemd. Er bestaan efficiënte methoden om de kansverdeling $P(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T | y_1, \ldots, y_T)$ over de toestanden $\boldsymbol{\theta}_t$ uit te rekenen onder de bovenstaande dynamica, en gegeven de waarnemingen die met deze toestanden verbonden zijn (zoals in de vorige subsectie).

In Hoofdstuk 4 beschouwen we een collectie van parallelle tijdreeksen. Iedere tijdreeks wordt gemodelleerd via toestanden $\boldsymbol{\theta}_{it}$, waarvan de dynamica wordt beschreven door de Kalmanfiltervergelijking. We introduceren tevens een 'gemiddeld' DLM, dat wordt gemodelleerd via toestanden $\mathbf{M}_t$. Iedere toestand $\mathbf{M}_t$ is verbonden met alle toestanden $\boldsymbol{\theta}_{it}$ op dezelfde tijd t. Dit model vertoont derhalve twee soorten afhankelijkheden: door de tijd heen, tussen opeenvolgende toestanden binnen één DLM, en tussen een gemiddelde toestand $\mathbf{M}_t$ en een parallelle toestand $\boldsymbol{\theta}_{it}$. Dit type model is een uitbreiding van het 'standaard' dynamisch hiërarchisch model (DHM). Het standaard DHM vertoont verbindingen tussen opeenvolgende gemiddelde toestanden $\mathbf{M}_t$, en tussen toestanden $\mathbf{M}_t$ en $\boldsymbol{\theta}_{it}$. Het bevat echter geen afhankelijkheden door de tijd heen, tussen toestanden $\boldsymbol{\theta}_{it}$ binnen de parallelle DLMs. De combinatie van afhankelijkheden door de tijd heen *en* tussen de toestanden $\mathbf{M}_t$ en $\boldsymbol{\theta}_{it}$ maakt de berekening van de posterior enorm complex. Het is in feite niet meer te doen binnen een acceptabele tijd. In Hoofdstuk 4 beschrijven we daarom twee benaderingen, die met het exacte model en met elkaar vergeleken zullen worden.

Zacht clusteren

Clustermethoden bestaan al heel lang, en in vele varianten. De meeste methoden nemen een collectie objecten (stukken fruit, complexe getallen, mensen, modellen) en sorteren hen netjes in een vastgesteld aantal groepen. Ieder object hoort dan bij exact één cluster. In dit proefschrift beschouwen we een 'zachter' alternatief, waarbij objecten, in plaats van een harde toewijzing

aan één cluster, een kansverdeling hebben die aangeeft 'hoe sterk' ze bij een willekeurig cluster horen.

In Hoofdstuk 5 wordt deze methode toegepast op modellen. We definiëren een afstandsmaat tussen modellen die gebaseerd is op modeluitvoer. Ieder cluster wordt gedefinieerd door zijn clustercentrum, en de toewijzingswaarschijnlijkheden voor ieder model hangen af van zijn afstand tot de bijbehorende centra. De belangrijkste rol van deze centra is echter het geven van een compacte representatie van het volledig modelensemble. De resultaten worden gebruikt om inzicht te verwerven in de werking van bootstrap-methodes, en om betekenisvolle clusters te vinden in een praktisch multitask learning-probleem.

Conclusies en discussie

In Hoofdstuk 2 laten we zien dat survival analysis baat heeft bij een Bayesiaanse benadering. De problemen van overfitten die voorkomen bij het origineel proportional hazards model worden in deze benadering voorkomen, en onze voorspellingen zijn beter dan die van het klassieke model. Beide modellen schatten een waarschijnlijkheid dat een nieuwe patiënt (één die niet betrokken was bij de training van het model) sterft na een zekere tijd t. Wanneer we deze schattingen vergelijken voor tijd $t_\dagger$, de waargenomen tijd van overlijden van die patiënt, dan is de geschatte waarschijnlijkheid onder ons model (gemiddeld) tot 80% hoger dan onder het klassieke model. Uit vergelijking van de drie benaderingen van de bijbehorende Bayesiaanse posterior, nl. MCMC sampling, een variationele benadering en de Laplace-benadering, blijkt dat enerzijds MCMC sampling de beste voorspellingen levert (tot 30% nauwkeuriger dan de andere twee benaderingen), maar anderzijds de variationele benadering geschikter is om de Bayesfactor uit te rekenen. Deze factor kan analytisch worden uitgerekend wanneer de posterior uitgedrukt wordt als een analytische functie, maar vereist een enorm aantal samples bij een numerieke benadering. De toepassing van de Bayesfactor stelde ons in staat meer dan 85% van de modelinvoer te elimineren, zonder enig verlies van functionaliteit. Sterker nog: het gereduceerde model, nu ongehinderd door 'irrelevante' invoer, presteert significant beter (tot 5%) dan het model met volledige invoer.

In Hoofdstuk 3 presenteren we een Bayesiaanse versie van multitask learning. De testresultaten worden (onder andere) behaald op een database die de losse verkoop van kranten in Nederland beschrijft. We laten zien dat multitask learning met één cluster (met een enkele Gaussische prior) beter voorspellingen levert dan het leren van singuliere taken, waar voor iedere taak afzonderlijk een model wordt geoptimaliseerd. Wanneer we de voorspelfout definiëren als het verschil tussen het voorspelde en het werkelijk verkochte aantal kranten, is de fout na multitask learning gemiddeld 10% kleiner dan na het leren van singuliere taken. Bayesiaans multitask learning verslaat (met 1%) ook de

niet-Bayesiaanse variant, waarbij alleen hard delen wordt geïmplementeerd: alle taken worden voorspeld door één MLP met een afzonderlijke uitvoer voor iedere taak. Het clusteren van taken (waarbij we een priorverdeling in de vorm van een som van Gaussische verdelingen implementeren) verbetert de prestaties op de krantendata niet. Het levert echter wel een betekenisvolle clusterverdeling van de taken op: verkoop op toeristische locaties en in kleine dorpjes worden toegewezen aan één cluster, verkoop in relatief grote steden in een ander. Hoofdstuk 3 demonstreert de twee voordelen van het clusteren van taken: ten eerste levert het betere voorspellingen op (soms al bij één cluster), en ten tweede biedt het, vanuit de clustertoewijzing van een taak, nieuwe inzichten in de karakteristieken van die taak.

Hoofdstuk 4 introduceert de combinatie van multitask learning en de analyse van tijdreeksen in de vorm van een dynamisch hiërarchisch model. Het resulterende model blijkt beter te presteren dan het standaard DHM waarop het gebaseerd is. Een mogelijke reden voor deze verbetering is zichtbaar in een grafiek van de gemiddelde toestanden als functie van de tijd: terwijl de gemiddeldes voor de parallelle tijdreeksen zich onder het oude model chaotisch gedragen, is de dynamica voor de onder het nieuwe model berekende gemiddeldes veel soepeler. Er worden in dit hoofdstuk twee benaderingen (variationeel en factorieel) van het volledig grafisch model bestudeerd. In termen van voorspellingskwaliteit presteren beide benaderingen even goed op de beschouwde data. De rekentijd die nodig is om de posteriorverdeling uit te rekenen groeit voor beide benaderingen lineair met het aantal parallelle taken, terwijl die groei bij exacte inferentie veel sterker is. De factoriële benadering is een beetje langzamer dan de variationele benadering, maar ook nauwkeuriger: uitgedrukt in termen van KL-divergentie benadert zij de exacte posterior beter. Dit laat zich echter niet vertalen naar nauwkeuriger voorspellingen, daar beide benaderingen al bijna even goed voorspellen als het exacte model. Er wordt bewezen dat onder beide benaderingen de uitgerekende gemiddeldes exact zijn.

In Hoofdstuk 5 presenteren we een methode om modellen te clusteren. We gebruiken deze methode om modelensembles te representeren door een klein aantal clustercentra. Het eerste modelensemble wordt verkregen door bootstrapping. We laten zien dat voorspellingen die gebaseerd zijn op slechts een paar clustercentra al even goed kunnen zijn als voorspellingen gebaseerd op het volledig ensemble. Het clusteren van modellen wordt verder gebruikt om de effecten van zowel 'bias'- als 'variance'-reductie in bootstrap-methodes te demonstreren. Het tweede ensemble wordt gevormd door het onafhankelijk leren van een collectie van parallelle taken (één of meerdere onafhankelijke modellen worden geleerd voor iedere taak). Hier wordt het clusteren van modellen gepresenteerd als een alternatieve vorm van meertaaksleren en het clusteren van taken. Er wordt aangetoond dat de taakclusters soortgelijke karakteristieken hebben als die in Hoofdstuk 4. Beide experimenten tonen dat het clusteren van modellen gebruikt kan worden als een effectief stuk ana-

lytisch gereedschap voor data mining.

De wetenschap schrijdt voort, maar is nooit voltooid. Hoewel het werk in dit proefschrift oplossingen levert voor sommige problemen, en verbeteringen in enkele velden, dient het ook te leiden tot nieuwe ideeën voor rivaliserende, alternatieve benaderingen. De survival analysis methode gepresenteerd in Hoofdstuk 2, bijvoorbeeld, heeft bewezen een goede benadering te zijn, maar er zijn nog verbeteringen mogelijk. Een alternatief model zou kunnen worden gevonden in het veld van de tijdreeksanalyse, aangezien iedere patiënt beschreven kan worden als een toestand die evolueert door de tijd, waarbij de overlevingskansen van de patiënt afhangen van deze toestand. Hoofdstuk 2 beschrijft verder nog een aantal bestaande, alternatieve benaderingen van survival analysis, waarvan enkele nog verbeterd kunnen worden middels een Bayesiaanse benadering.

De verdelingen die in Hoofdstuk 3 en 4 gebruikt worden zijn enkele Gaussische verdelingen. Hoewel zulke verdelingen computationeel zeer aantrekkelijke eigenschappen hebben, zouden andere verdelingen zeer interessante aspecten kunnen toevoegen, en daarmee geschikter kunnen zijn voor sommige taken. Het model dat in Hoofdstuk 4 gepresenteerd wordt bevat zuiver continue toestanden. In verder onderzoek zou dit model kunnen worden uitgebreid door discrete toestanden toe te staan. Verder zouden Hoofdstuk 3 en 4 kunnen worden gecombineerd om het clusteren van taken een plek te geven binnen het dynamisch hiërarchisch model. In dit geval zou ieder modelcluster de vorm hebben van het grafisch model omschreven in Hoofdstuk 3: een set parallelle dynamische lineaire modellen die individueel door de tijd verbonden zijn, en allemaal verbonden met een gemeenschappelijke, gemiddelde DLM.

Verdere ontwikkelingen betreffende het idee van het clusteren van modellen zouden gevonden kunnen worden in de toepassingen. Dit proefschrift heeft haar gebruik bij bootstrapping-methodes en multitask learning gedemonstreerd. Andere mogelijkheden zijn het clusteren van samples uit een Bayesiaanse posteriorverdeling, of detectie van overeenkomsten binnen een ensemble van modellen met verschillende architecturen.

Curriculum Vitae

Bart Bakker werd geboren op 26 juli 1975 te Zevenaar. Na de middelbare school begon hij met de studie Natuurkunde aan de Katholieke Universiteit Nijmegen. Al snel raakte hij zeer geïnteresseerd in zowel de theoretische als de informatische kant van de natuurkunde. De richting Informatische Fysica leek beide goed te combineren, en hierin specialiseerde hij zich dan ook. Tijdens zijn studie heeft hij veel tijd besteed aan het begeleiden van studenten bij werkcolleges en computerpractica. Meer tijd besteedde hij echter nog aan bestuurs- en commissiewerk voor zijn favoriete studievereniging, Marie Curie. In 1998 studeerde hij af op het gebied van de Biofysica, onder begeleiding van dr. Pieter Medendorp en prof. dr. C.C.A.M. Gielen. Tijdens zijn afstudeerstage maakte hij kennis met het vakgebied Neurale Netwerken, waarop een paar deuren verder onderzoek werd gedaan. De zeer sterke wisselwerking tussen wiskunde, informatica en theoretische natuurkunde binnen dit vakgebied was voor hem meer dan genoeg reden om te informeren naar een mogelijke promotieplek. Deze bleek gecreëerd te kunnen worden, en al snel begon hij als OiO op het STW project 'Knowledge representation with neural networks', onder begeleiding van dr. Tom Heskes en prof. dr. Gielen. Het resultaat vindt u in dit proefschrift. Na zijn werkzaamheden als OiO is hij aangenomen als post-doc bij de afdeling 'Man-Machine Interfaces' van het Philips Research Lab in Aken.